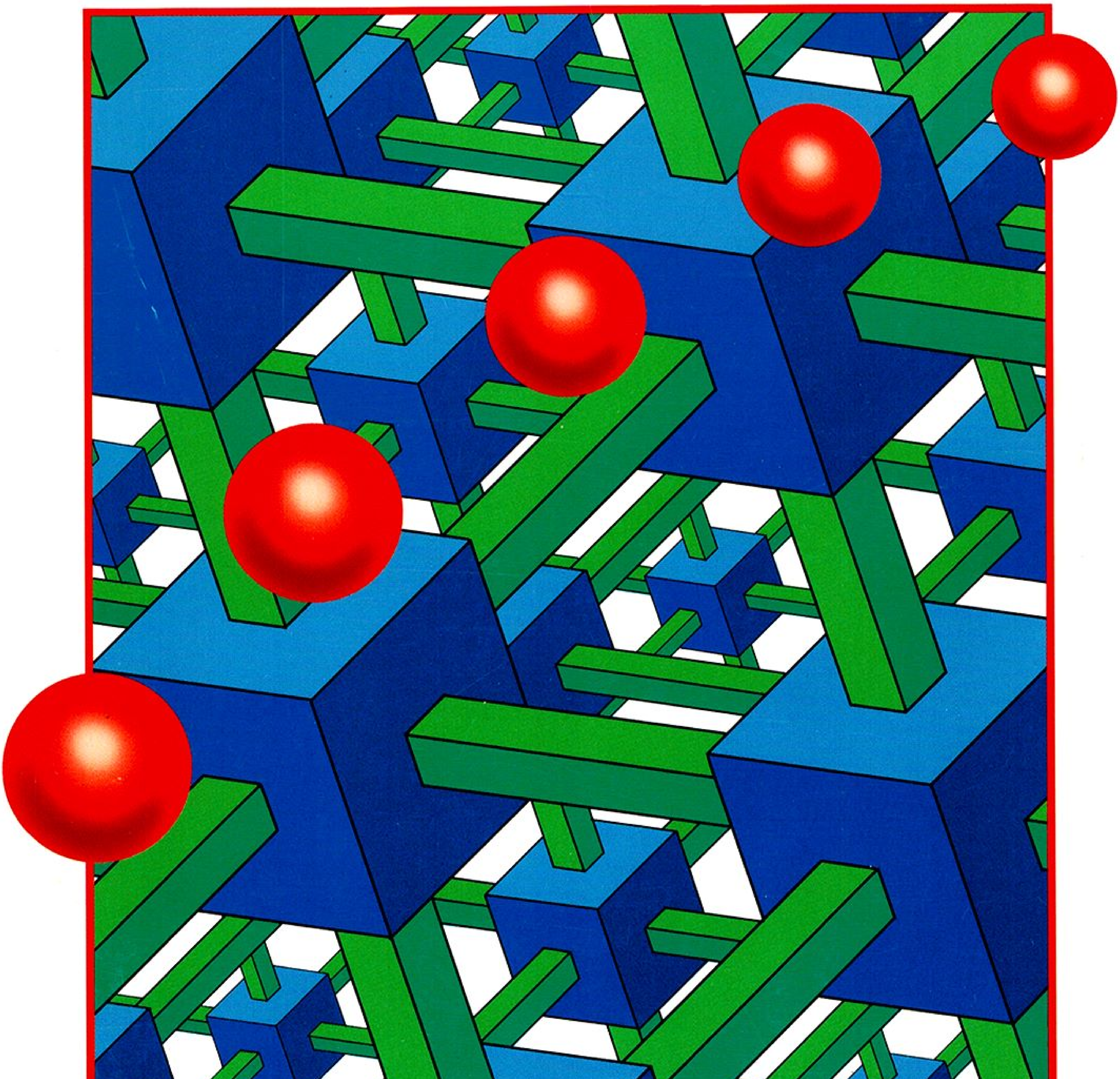
MSX-Datapack

エムエスエックス・データパック

Volume 1

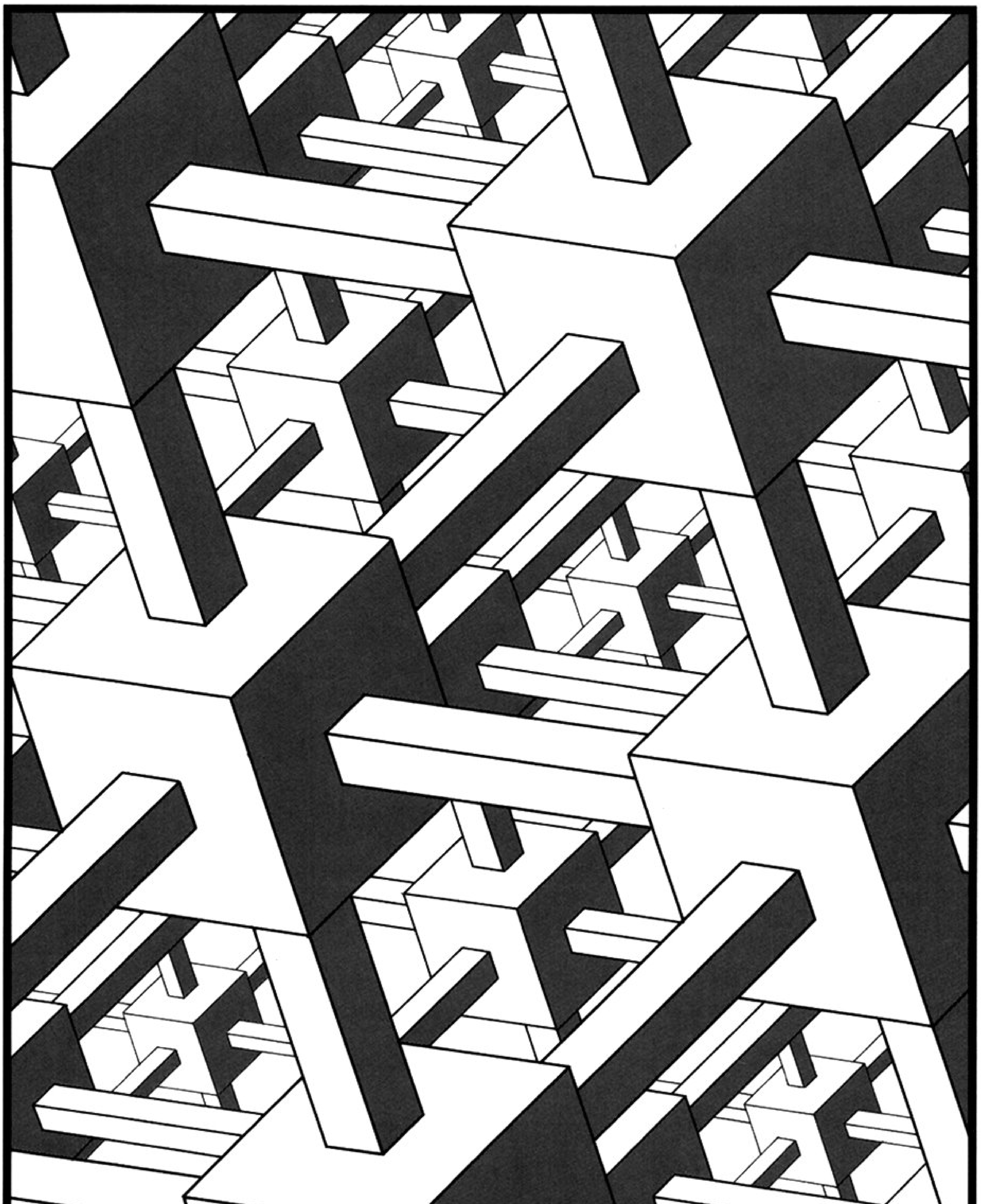


ASCII

MSX-Datapack

エムエスエックス・データパック

Volume 1



ASCII

はじめに

このたびは、「MSX-Datapack」をお買い上げいただきまして、誠にありがとうございます。「MSX-Datapack」はMSX、MSX2、MSX2＋の機能をハードウェアとソフトウェアの両面から解説したマニュアルとサンプルプログラムのパッケージです。

MSXホームパーソナルコンピュータは、異なるメーカーのコンピュータ間での互換性を実現する試みとして私どもアスキーから提唱しました。

当初、MSXはローエンドのホームパーソナルコンピュータとして企画されましたが、その柔軟性、拡張性に富んだ構成とMSX-DOSの組み合わせにより、ゲームやプログラム開発、計測器の制御など様々な応用ができます。

しかし、MSXの柔軟性と拡張性は使い方を誤ると、MSXの互換性を損なうことになりかねません。MSXの互換性を守ってゆくために、ハードウェアやソフトウェアを作成する方は、このパッケージの内容をルールとして守って下さい。

このパッケージには、以下のものが含まれています。

■MSX-Datapack マニュアル　1冊

■MSX-Datapack サンプルディスク　1枚
(3.5-2DD フロッピーディスク)

ご注意

1. このソフトウェアおよびマニュアルの内容の一部または全部を無断で使用、複製することはできません。

2. このプログラムは個人として利用するほかは、著作権上、株式会社アスキーに無断で使用することはできません。

3. 製品の仕様は将来予告なく変更することがありますが、製品登録カードをご返送いただいた方にはそのつどご案内いたします。

4. 製品の内容については万全を期しておりますが、万一ご不審な点や誤り、記載のもれなどお気づきの点がございましたら弊社までご連絡下さい。

5. このソフトウェアおよびマニュアルを運用した結果の影響については、4項にかかわらず責任を負いかねますのでご了承下さい。

このマニュアルの読み方

このマニュアルの構成

このマニュアルは以下のように構成されています。

Volume 1

第1部　ハードウェア

MSXのハードウェア仕様を解説します。

第2部　システムソフトウェア

ブートシーケンスや割り込み、BASICの仕様や内部ルーチンのなどを解説します。

第3部　MSX-DOS

MSX-DOSの操作法やファンクションコールの使用法などを解説します。

第4部　VDP

V9938とV9958の機能と使用法を解説します。

Volume 2

第5部　スロットとカートリッジ

第6部　標準的な周辺装置のアクセス

第7部　オプションの周辺装置のアクセス

APPENDIX

BIOSやワークエリアなどの一覧表です。

このマニュアルの表記

1. MSX のバージョン

MSX1	MSX BASIC version 1.0 を搭載した MSX
MSX2	MSX BASIC version 2.0 を搭載した MSX
MSX2+	MSX BASIC version 3.0 を搭載した MSX
MSX	MSX1、MSX2、MSX2+の総称

2. キー

2つのキーが「+」で結ばれているときは、左側のキーを押しながら、右側のキーを押します。例えば、[CTRL] + [STOP] は、[CTRL] キーを押しながら [STOP] キーを押すことを示します。

3. BIOS エントリ、内部ルーチン

MSX2、MSX2+では BIOS が MAIN ROM と SUB ROM に分割されています。これを区別するため、以下のように表記します。

CHGET(009FH / MAIN)	MAIN ROM の 009FH 番地
REDCLK(01F5H / SUB)	SUB ROM の 01F5H 番地

4. ワークエリア、フック

【VALTYP(F663H, 1)】	F663H 番地の 1 バイトを使用
【BUF(F55EH, 258)】	F55EH 番地以降の 258 バイトを使用

5. その他、必要に応じて解説します。

目次

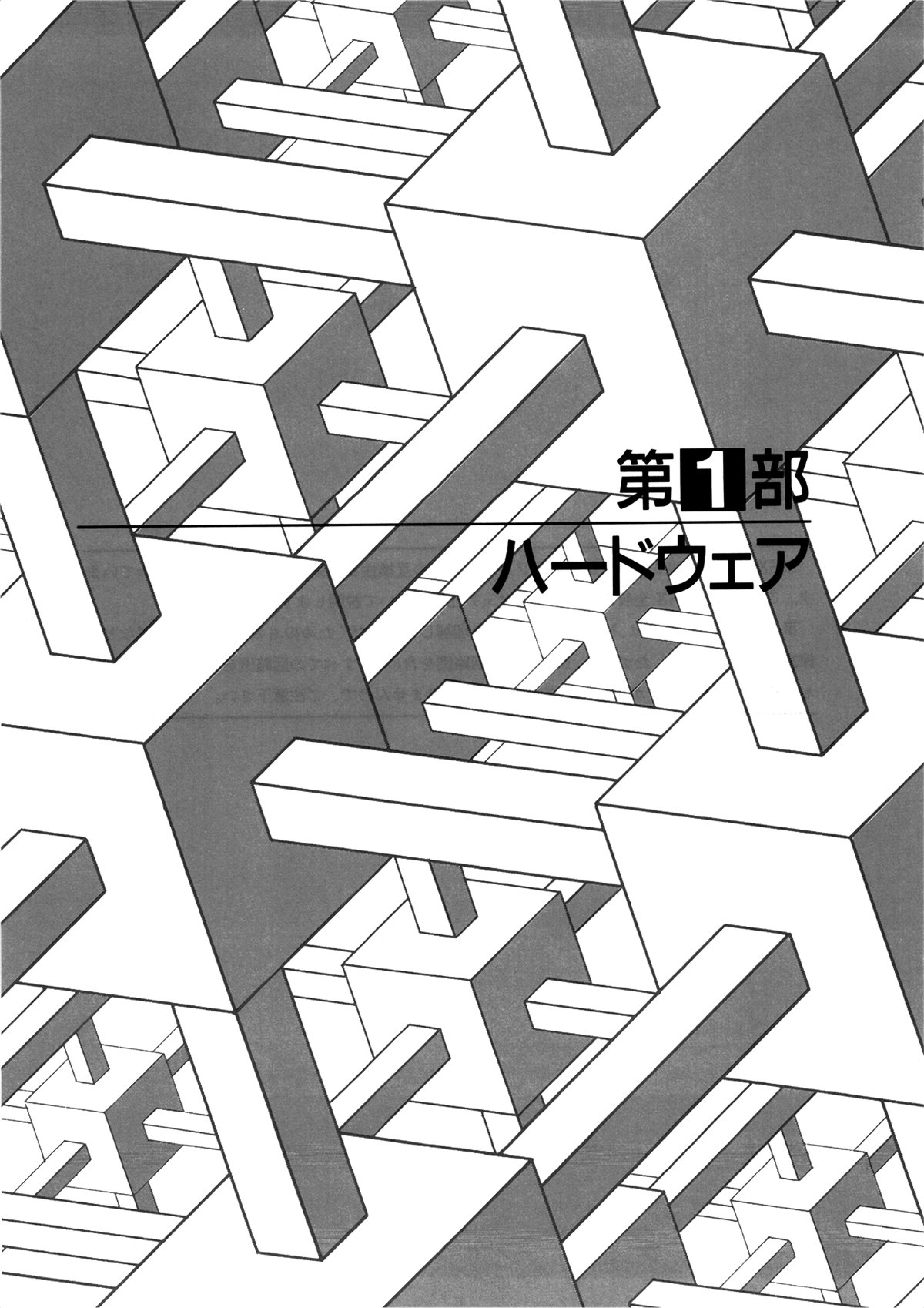
<MSX-Datapack　Vol.2　目次>

第 7 部　オプションの周辺機器

APPENDIX

第1部

ハードウェア



MSXは将来に渡るハードウェアとソフトウェアの互換性と拡張性を考え、仕様を定めています。ここでは、MSX全般に渡るハードウェア仕様について説明します。

第1部にある回路例は、MSXの動作論理を理解していただくためのもので、タイミングなどの保証はありません。したがって、MSX2+の回路例を含めて、すべての回路例をそのまま実現しても、そのハードウェアが正常に動作する保証はありませんので、ご注意下さい。

1章 概略仕様

表 1.1 MSX の概略仕様一覧

		MSX2+	MSX2	MSX
CPU		Z80A 相当品（クロック 3.579545MHz±1%）		
メモリ	ROM	80KB (MSX BASIC ver3.0) MAIN ROM 32KB SUB ROM 16KB 漢字ドライバ 16KB 単漢字変換 16KB	48KB (MSX BASIC ver2.0) MAIN ROM 32KB SUB ROM 16KB	32KB (MSX BASIC ver.1.0) MAIN ROM 32KB
	RAM	64KB 以上	64KB 以上	8KB 以上
	VRAM	128KB	64KB または 128KB	16KB
VDP		V9958 相当品	V9938 相当品	TMS9918A 相当品
	解像度(最大)	512×212 (ノンインタレース時) 512×424 (インタレース時)	512×212 (ノンインタレース時) 512×424 (インタレース時)	256×192 なし
	表示色(最大)	19268 色	256 色	16 色
	ハードウェアスクロール	縦・横	縦	なし
CMT カセットインターフェイス		FSK 方式 1200・2400bps		
サウンド	PSG	AY-3-8910 相当品		
	FM 音源	MSX-AUDIO（オプション）		
		MSX-MUSIC（オプション）	FM-PAC で対応	なし
キーボード		英数、ひらがな、カタカナ、グラフィック機能対応 JIS 配列・50 音配列対応		
フロッピーディスクフォーマット		3.5 インチ 1DD（2DD、2HD、ハードディスクはオプション） MS-DOS 準拠		
プリンタ		8 ビットパラレル セントロニクスインターフェイス準拠		（オプション）
カートリッジスロット		カートリッジバス ROM カートリッジ、拡張バスに対応するスロットを 1 つ以上持つ		
ジョイスティック		2		1 または 2
漢字機能	漢字 ROM	第一水準 第二水準(オプション)	(オプション) (オプション)	(オプション)
	漢字入力	単漢変換(MSX-JE 対応)	アプリケーションで対応	
リアルタイムクロック IC		RP5C01 相当品		なし

2章 主要ユニット

2.1 LSI

CPU	Z80A 相当品	
	クロック 3.579545MHz（テレビ色副搬送波周波数）	
	M1 サイクルで 1WAIT 挿入	
VDP	MSX	TMS9918A 相当品
	MSX2	V9938 相当品
	MSX2＋	V9958 相当品
PSG	AY-3-8910 相当品	
FM 音源	MSX-AUDIO（Y8950 相当品）	
	MSX-MUSIC（YM2413 相当品）	
PPI	i8255 相当品	

2.2 メモリ

2.2.1 使用メモリ

ROM	MSX	MSX BASIC ver1.0	32KB
	MSX2	MSX BASIC ver2.0	48KB

		MAIN ROM 32KB、SUB ROM	16KB
	MSX2＋	MSX BASIC ver3.0	80KB
		MAIN ROM 32KB、SUB ROM	16KB
		漢字ドライバ	16KB
		単漢変換	16KB
RAM	MSX	8KB以上	
	MSX2	64KB以上	
	MSX2＋	64KB以上	

2.2.2 メモリマッパー

1. メモリマッパーの概要

MSXではCPUのメモリ空間をスロットという空間（「3.9 スロット」参照）に割り付けます。

メモリマッパーとは、MSXの任意のスロットに装備することができ、スロットのメモリ空間の論理アドレスをメモリの物理アドレスに変換し、1つのスロットに接続できるメモリ空間を最大で4Mバイトまで拡張する装置です。

スロットのメモリ空間の各ページ（16Kバイト×4ページ）は、最大で256ページ（1ページは16Kバイト）の任意のマッパーセグメントに割り当てられます。マッパーセグメントとスロットのメモリ空間の物理ページの対応は、マッパーセグメント選択レジスタの内容によって決められます。

マッパーセグメント選択レジスタはI/OアドレスのFCH番地からFFH番地までの4バイトに置かれ、それぞれのレジスタは、CPU空間の物理ページ0〜3までに対応します。

マッパーセグメント選択レジスタは書き込み専用で、1つのスロットに対して1組用意することができます。このレジスタに値を書き込むと、他のスロットのマッパーセグメント選択レジスタにも同時に値が書き込まれます。その結果、すべてのスロットのマッパーセグメントが切り換わるので注意して下さい。「3.9 スロット」を参照して下さい。

メモリマッパーはMSX2、MSX2＋のオプション機能です。メモリマッパーに接続されるメモリは最小で64Kバイトとします。

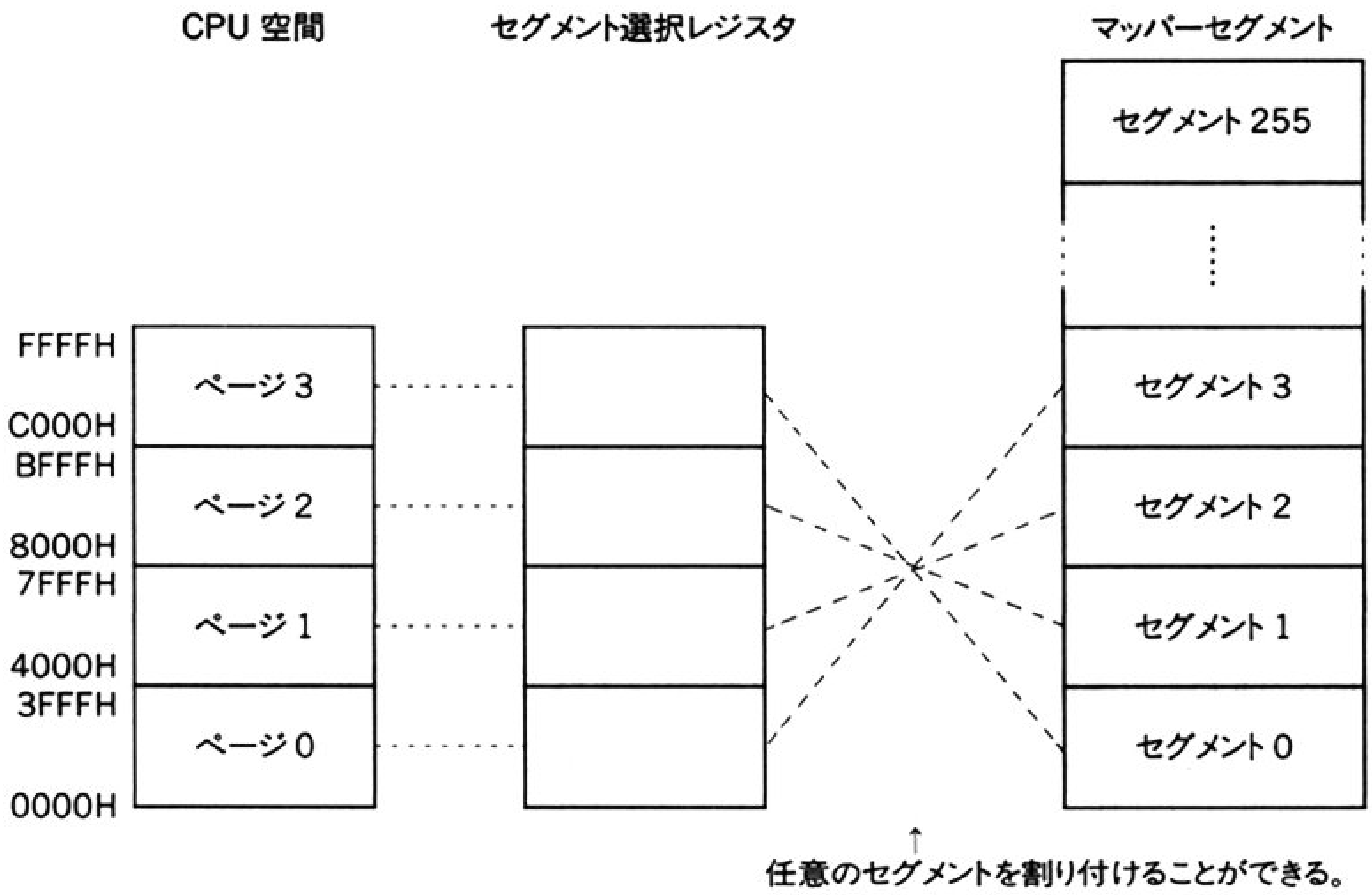


図 1.1　メモリマッパーの論理的構成

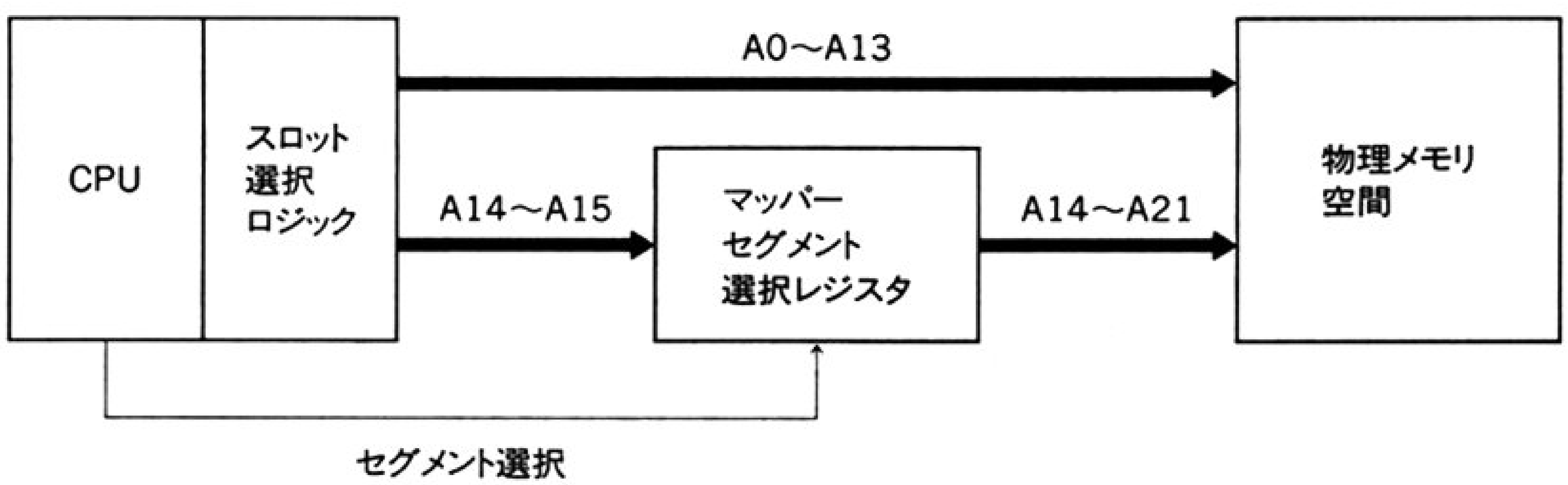


図 1.2　メモリマッパーの物理的構成

2. マッパーセグメント選択レジスタの初期化

MSX2、MSX2+システムは、リセット時にマッパーセグメント選択レジスタを下記の値で初期化します。メモリマッパー有無の検出は行いません。マッパーセグメント選択レジスタの初期化は、I/Oポートにデータを書くだけの処理なので、メモリマッパーの機構を持たないハードウェアに対して影響はありません。

I/Oアドレス	初期化データ	CPU 空間のページ	マッパーセグメント
FCH	03H	0	3
FDH	02H	1	2
FEH	01H	2	1
FFH	00H	3	0

マッパーセグメント選択レジスタの初期化後、システムはRAMのサーチ、ROMのサーチなどの初期化を行います。詳しくは、「第2部　システムソフトウェア」を参照して下さい。

海外版のMSX BASIC version 2.1は、メモリディスク機能のためにメモリマッパーを使用します。また、MSX-DOS version 2はメモリマッパーが必須となります。その他のBASICやMSX-DOS version 1はメモリマッパーを使用しません。

3．メモリマッパーの使用方法

メモリマッパーの容量と実装されているスロット（複数のスロットに実装されていることもある）は、機種によって異なりますので、アプリケーションソフトウェアが調べる必要があります。

ただし、MSX-DOS2の環境では、メモリマッパーはすべてDOSが管理するので、MSX-DOS2のメモリアクセスの規則（拡張BIOSコール）にしたがって下さい。

注　意

- メモリマッパーを使用したアプリケーションプログラムがBASICやDOSに制御を戻すときには、マッパーセグメント選択レジスタを初期状態に戻さなければなりません（DOS2使用時を除く）。
- メモリマッパーを必要とする商用ソフトウェアは、メモリマッパーが必要である旨と必要とする容量がユーザーにわかるよう、「マッパーRAM（xxxKB）必要」などと明記して下さい。
- マッパーセグメント選択レジスタの長さは、実装されているメモリ容量より大きくてもかまいません。たとえば、512Kバイトまで扱えるように5ビットのマッパーセグメント選択レジスタがあり、実装されているメモリが64Kバイトの機種もありえます。
- 1つのMSXシステムに複数のマッパーRAMが実装されていた場合、マッパーセグメント選択レジスタを書き換えると、他のスロットのマッパーRAMも切り替わります。
- DOS2の環境では、メモリマッパーはDOS2が管理しています。したがって、アプリケーションプログラムがマッパーセグメント選択レジスタを直接操作するときは、DOS2のメモリ管理の状況を十分把握していないと、システムのクラッシュ（暴走）を招きます。

4．メモリマッパー回路例

以下に、メモリマッパーの回路例を示します。この回路例は、メモリマッパーのロジックを理解する上で参照して下さい。

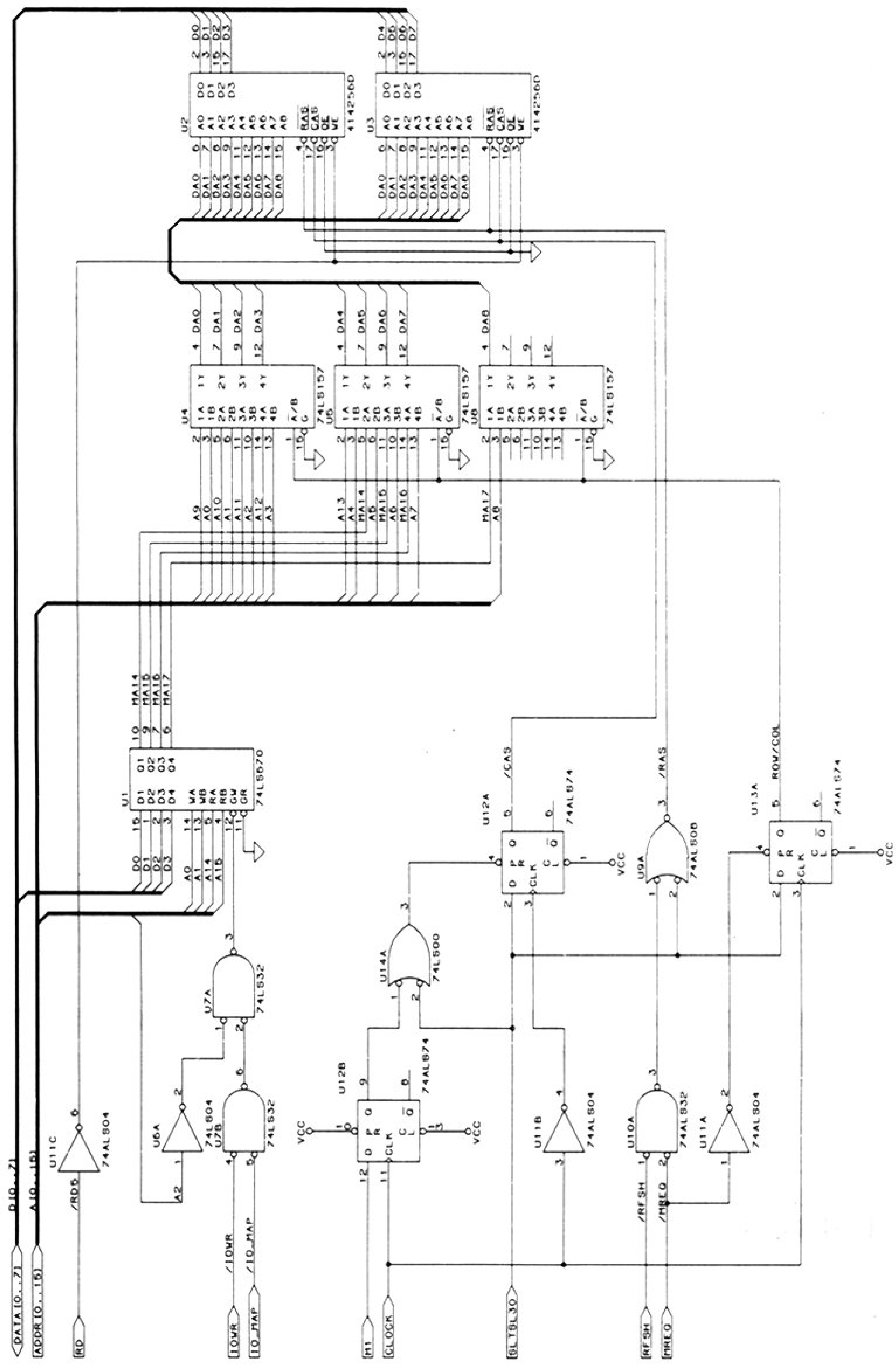


図 1.3　メモリマッパー回路

2.3 割り込み

NMI　使用しません。

INT　VDPおよびカートリッジバスからの外部割り込みです。

MSX BASICインタープリタは、VDPからの60Hz（NTSC方式）または50Hz（PAL・SECAM方式）の信号を割り込みとして使用します。割り込みモードは1(38H番地に分岐）です。

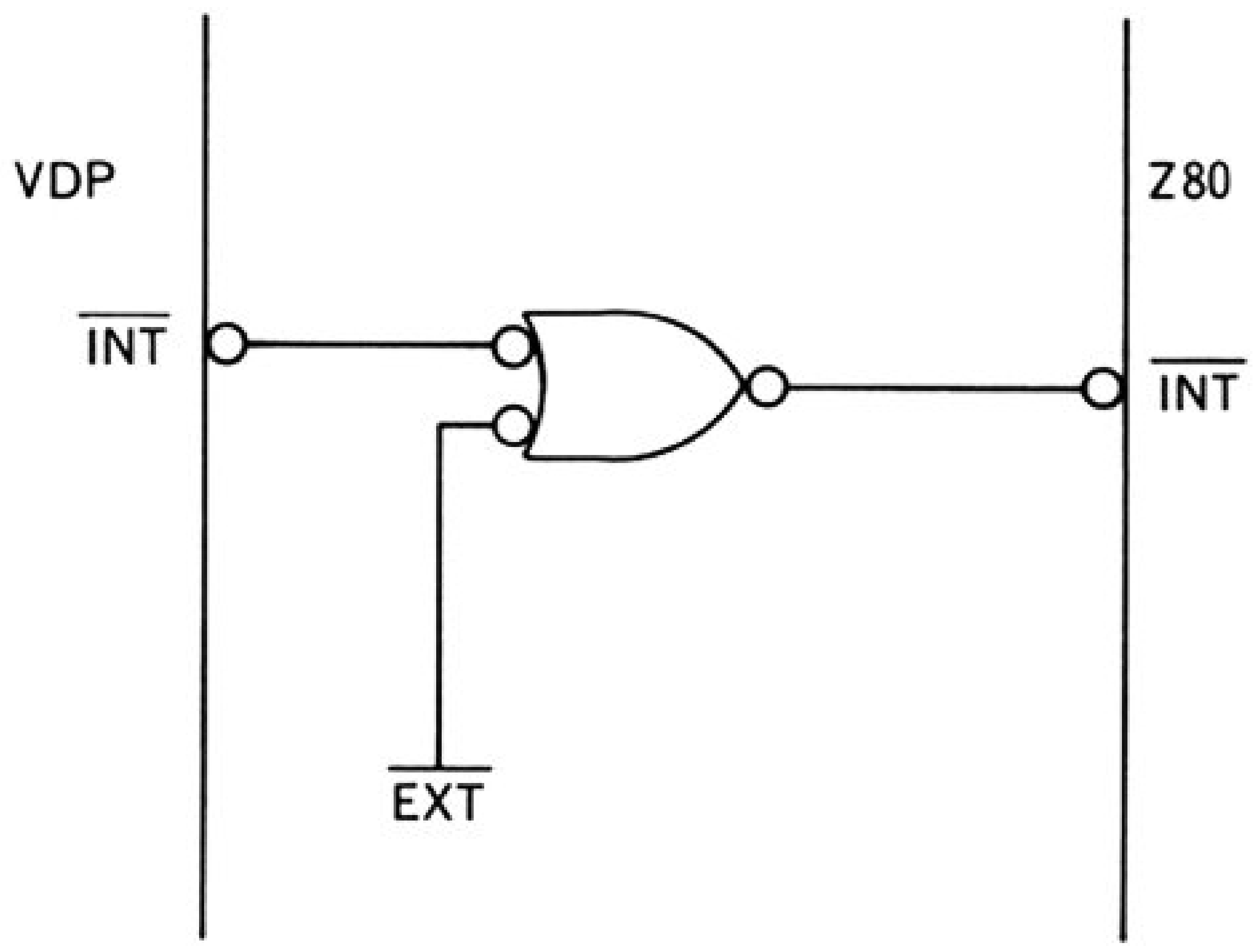


図1.4　割り込み回路例

注　意　MSX-DOSでは、66H番地（NMIエントリアドレス）がMSX-DOSのFCBデータエリアとして使用されるため、NMIは使用できません。

2.4 画面表示

使用IC	MSX	TMS9918A相当品
	MSX2	V9938相当品
	MSX2＋	V9958相当品

文字構成　英数文字＋ひらがな＋カタカナ＋グラフィック記号　256種

8×8ドット（英数文字とカタカナは右の2ドットを使用しない）

JIS第一水準漢字2965字（MSX、MSX2ではオプション）

	JIS 第二水準漢字 3384 字（オプション）	
最大表示色	MSX	16 色
	MSX2	256 色
	MSX2+	19268 色
スプライト	MSX	32 枚のスプライト。同一水平線上に 4 枚まで表示可能。
	MSX2 MSX2+	32枚のスプライト。同一水平線上に8枚まで表示可能。1ライン毎に色を変えることができる。
画面表示モード	「第 4 部 VDP」を参照して下さい。	

下位4ビット ＼ 上位4ビット	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		π		0	@	P		p	♠			—	タ	ミ	た	み
1	月	┴	!	1	A	Q	a	q	♥	あ	。	ア	チ	ム	ち	む
2	火	┬	"	2	B	R	b	r	♣	い	「	イ	ツ	メ	つ	め
3	水	┤	#	3	C	S	c	s	◆	う	」	ウ	テ	モ	て	も
4	木	├	$	4	D	T	d	t	○	え	、	エ	ト	ヤ	と	や
5	金	┼	%	5	E	U	e	u	●	お	・	オ	ナ	ユ	な	ゆ
6	土	│	&	6	F	V	f	v	を	か	ヲ	カ	ニ	ヨ	に	よ
7	日	─	'	7	G	W	g	w	ぁ	き	ァ	キ	ヌ	ラ	ぬ	ら
8	年	┌	(	8	H	X	h	x	ぃ	く	ィ	ク	ネ	リ	ね	り
9	円	┐	)	9	I	Y	i	y	ぅ	け	ゥ	ケ	ノ	ル	の	る
A	時	└	*	:	J	Z	j	z	ぇ	こ	ェ	コ	ハ	レ	は	れ
B	分	┘	+	;	K	〔	k	{	ぉ	さ	ォ	サ	ヒ	ロ	ひ	ろ
C	秒	╳	,	<	L	¥	|	¦	ゃ	し	ャ	シ	フ	ワ	ふ	わ
D	百	大	−	=	M	〕	m	}	ゅ	す	ュ	ス	ヘ	ン	へ	ん
E	千	中	・	>	N	∧	n	～	ょ	せ	ョ	セ	ホ	゛	ほ	
F	万	小	／	?	O	—	o		っ	そ	ッ	ソ	マ	゜	ま	

図 1.5　表示文字コード一覧

2.5 キーボード

配列	英数字	JIS 標準配列
	カナ	JIS 標準配列、五十音配列 （ジャンパによる切り換え）
キースキャン	ソフトウェアによる	
キートップ数	最大 88 個	
キーマトリックス図	（カナは五十音配列。JIS 標準配列は次ページ参照）	

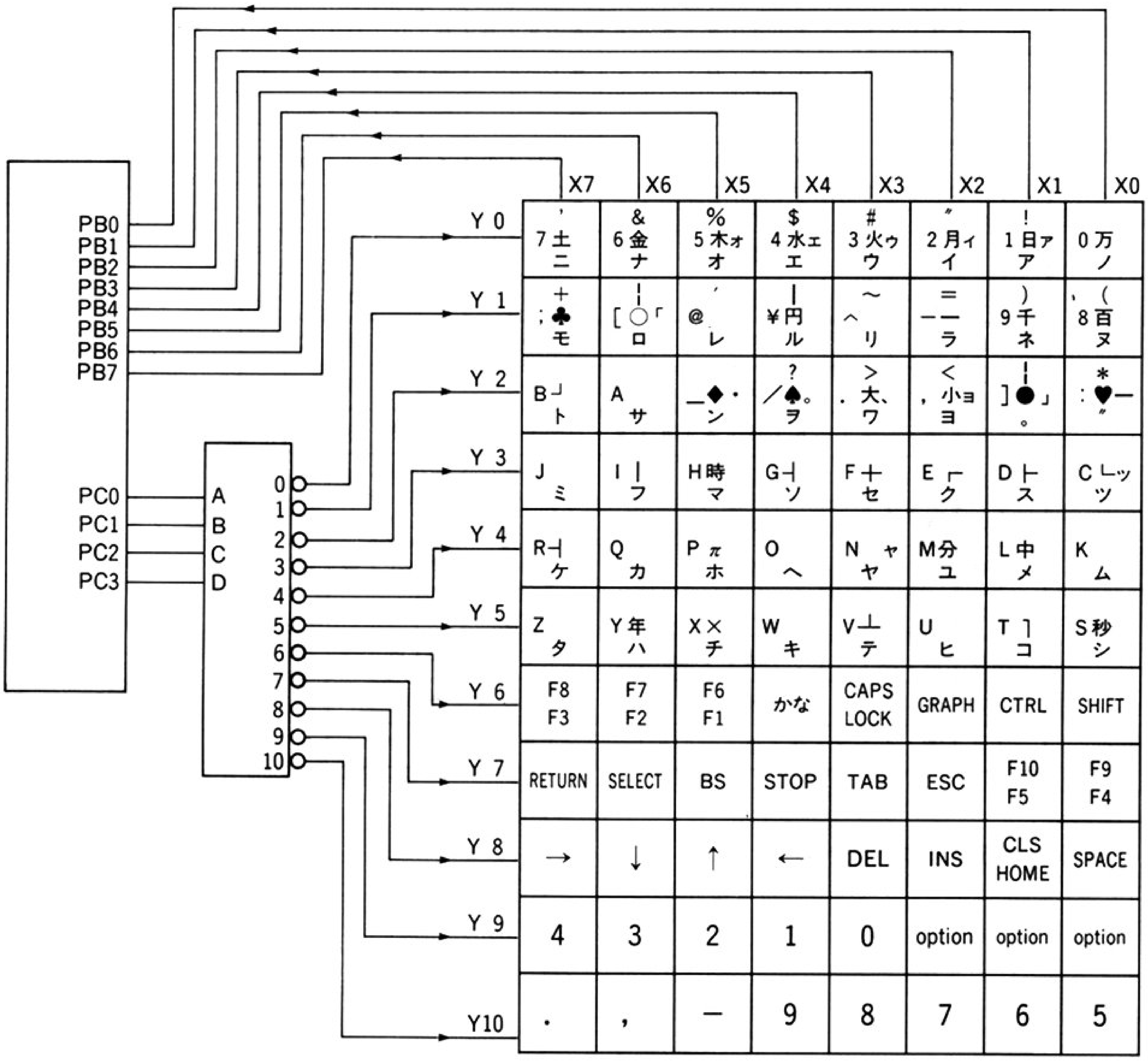


図 1.6　キーのマトリックス

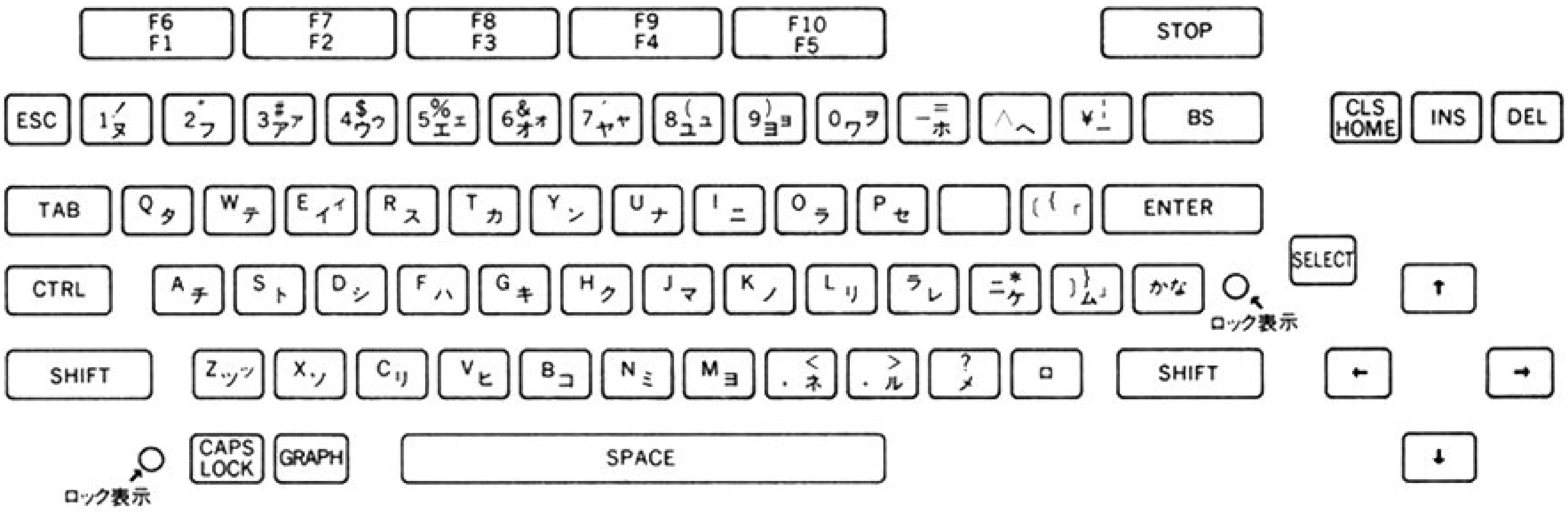


図 1.7　JIS 配列キーボード

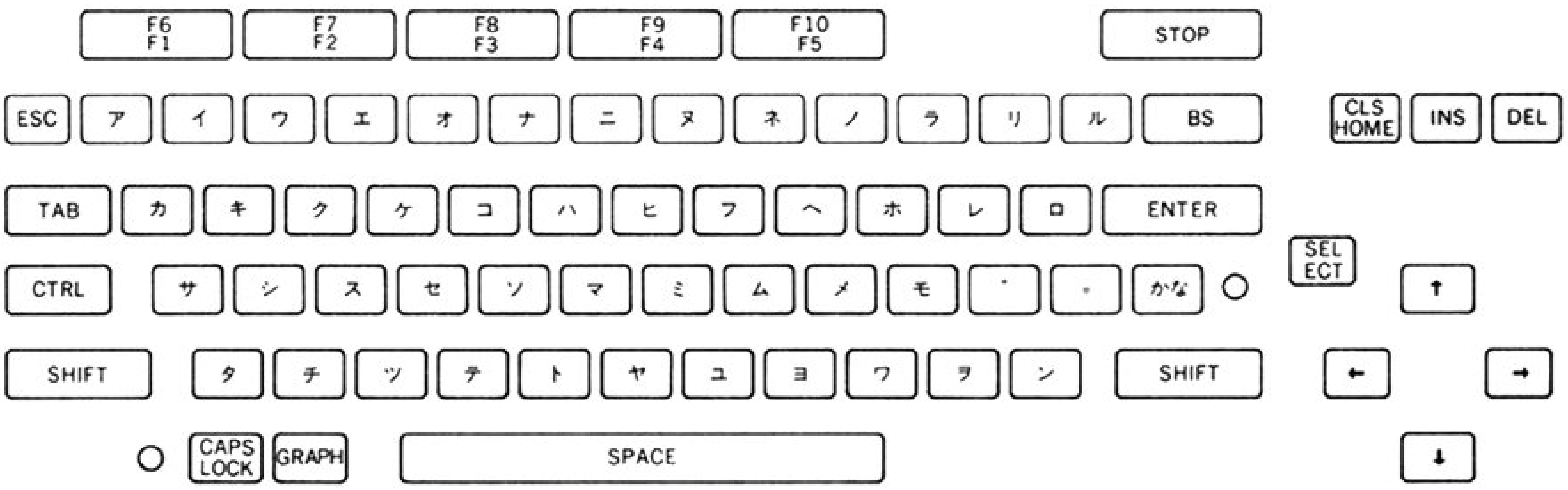


図 1.8　五十音配列キーボード

2.6 サウンド

2.6.1 音源 LSI と仕様

PSG

LSI	AY-3-8910 相当品　クロック 1.7897725MHz
音階	8 オクターブ（3 重和音出力可）
同時発生数	3 音＋ノイズ
特殊効果音機能	あり
ソフトウェアサウンド機能	1 bit ポートによる音出力

出力信号レベル	−5dbm
コネクタ	RCA 2ピン

MSX-AUDIO

LSI	Y8950相当品
音階	8オクターブ
同時発生数	9音または6音+5リズム
オペレータ数	2
ADPCM機能	あり
ADPCM用RAM	32KB〜256KB
ミュージックキーボード接続端子	あり
音声入力端子	マイクレベル　ミニジャック
音声出力端子	ラインレベル　RCA 2ピン

MSX-MUSIC

LSI	YM2413相当品
音階	8オクターブ
同時発生数	9音または6音+5リズム
オペレータ数	2
ADPCM機能	なし
ミュージックキーボード接続端子	なし
音声入力端子	なし
音声出力端子	RCA 2ピン

2.6.2 音声関連機能の内部接続

MSXの音声出力端子には、内部音源とオプション音源の音声出力がミキシングして出力されます。

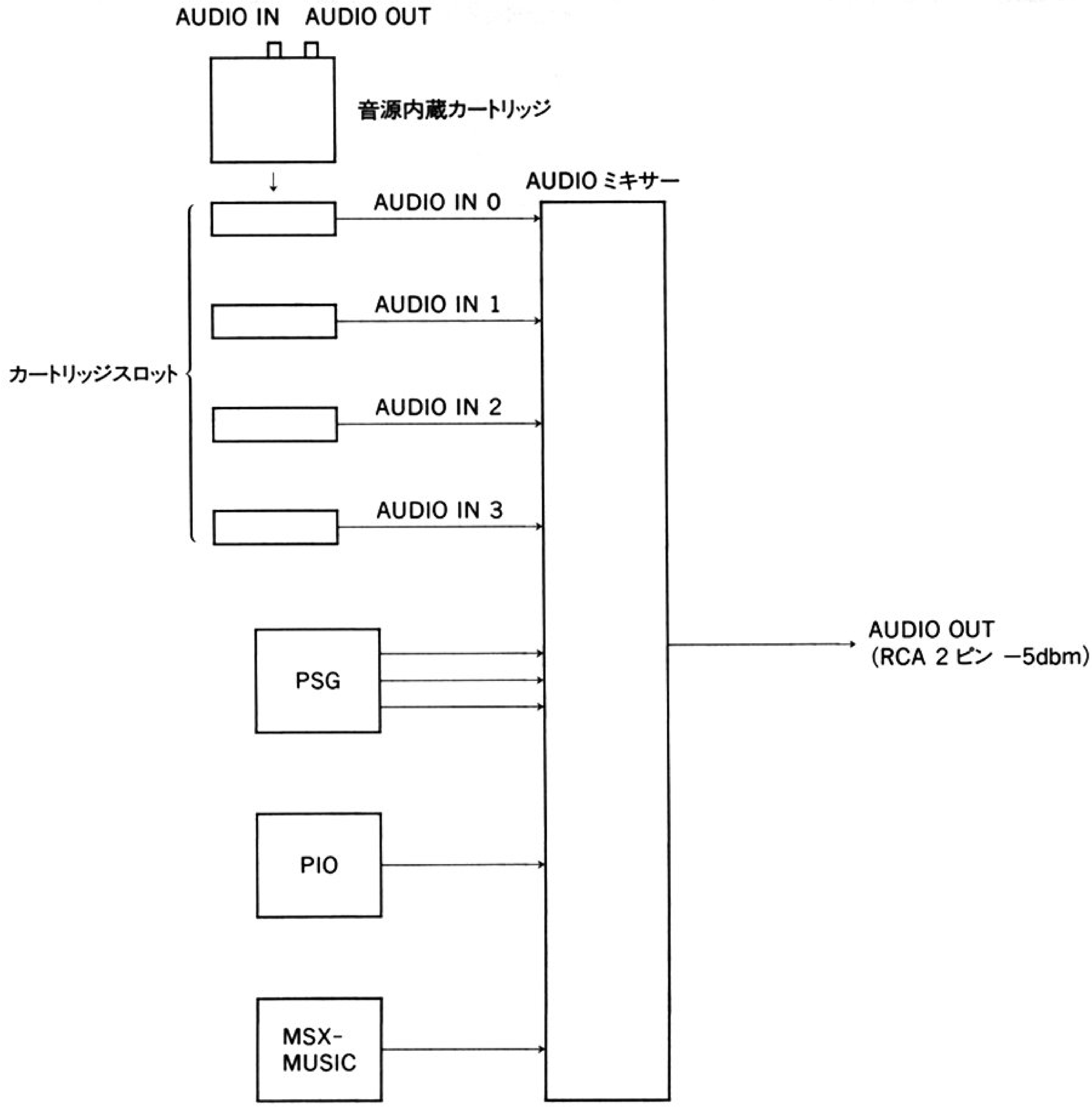


図 1.9 MSX の音声出力

3章 インターフェイス

3.1 カセットインターフェイス

入力	カセット側のイヤホン端子に接続
出力	カセット側のマイクロホン端子に接続
同期方式	調歩同期（非同期）方式
転送レート	1200bps　(1200Hz-1 波「0」、4800Hz-2 波「1」) デフォルト 2400bps　(2400Hz-1 波「0」、4800Hz-2 波「1」) ソフトウェア切り換え 2400bps はレコーダーの機種指定が必要（各メーカーで対応）
変調方式	FSK 方式
復調方式	ソフトウェアコントロール。BIOS ルーチンを使用することで、ボーレートは自動的に選択される。
リモート機能	あり
コネクタ	DIN45326（8 ピン）
SAVE レベル	SAVE 回路の定数は、以下の仕様を満たすように設定される。 出力レベル　　−45dBm±5dBm（0dBm = 0.775V） 1200Hz 信号時に 22mVp-p～7mVp-p の振幅が必要

端子番号	信号名	方 向	ピンコネクション
1	GND		
2	GND		
3	GND		
4	CMTOUT	出力	
5	CMTIN	入力	
6	REMOTE+	出力	
7	REMOTE−	出力	
8	GND		

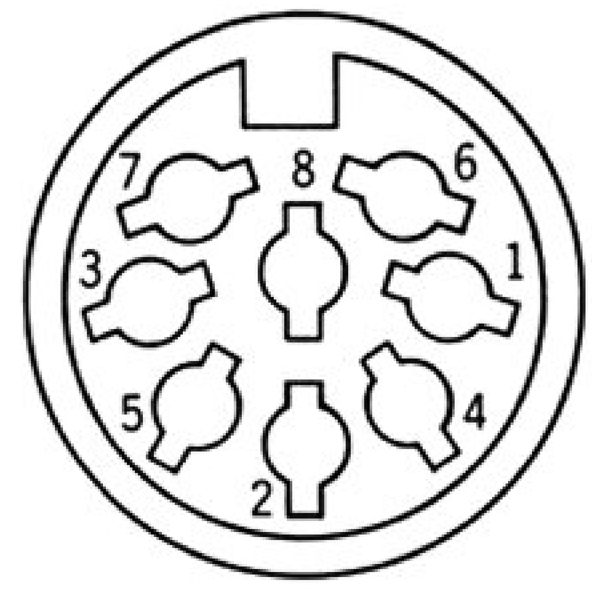


図 1.10　カセットインターフェイスの信号線

3.2 フロッピーディスクインターフェイス

フロッピーディスクインターフェイスのシステムソフトウェアは 4000H〜7FFFH の 16K バイトの ROM 内に置かれ、以下のモジュールを含みます。

MSX-DOS カーネル
MSX Disk BASIC
DISK I/O ドライバ（各メーカーが作成）

インターフェイスのハードウェアについては、特に規定を設けません。各ディスクドライブのメーカーによって作成される I/O ドライバがハードウェア上の差異を吸収します。

また、ディスクドライブはメディアの交換をハード的に検出できるディスク装置を推奨します。ディスクの入れ換えが検出できれば、ディスクのアクセスに要求される時間が短くなります。ハード的にフロッピーディスクの入れ換えが検出できない場合は、I/O ドライバは一定時間アクセスがないと、フロッピーディスクが交換されたものとみなします。そのため、次のディスクアクセスの際は、ディスクの入れ換えがないときでも再度ファイルの情報（FAT のセクタ 1 など）を読むので時間がかかります。

フロッピーディスクのフォーマットは以下のとおりです。

表1.2 フロッピーディスクのフォーマット

メディア	記録密度	フォーマット	備 考
8 inch*	単密度	128 Bytes/Sector	MS-DOS コンパチブル
8 inch*	倍密度	1024 Bytes/Sector	〃
5.25 inch*	単密度	512 Bytes/Sector	〃
3.5 inch	CFD	512 Bytes/Sector	〃

*商品化はされていません。

3.3 プリンタインターフェイス

規格	8ビットパラレル BUSYとSTROBE信号によるハンドシェイク
レベル	TTL
コネクタ	アンフェノール14ピン（本体側メス）

ピン番号	信号名
1	PSTB
2	PDB0
3	PDB1
4	PDB2
5	PDB3
6	PDB4
7	PDB5
8	PDB6
9	PDB7
10	NC
11	BUSY
12	NC
13	NC
14	GND

ピンコネクション

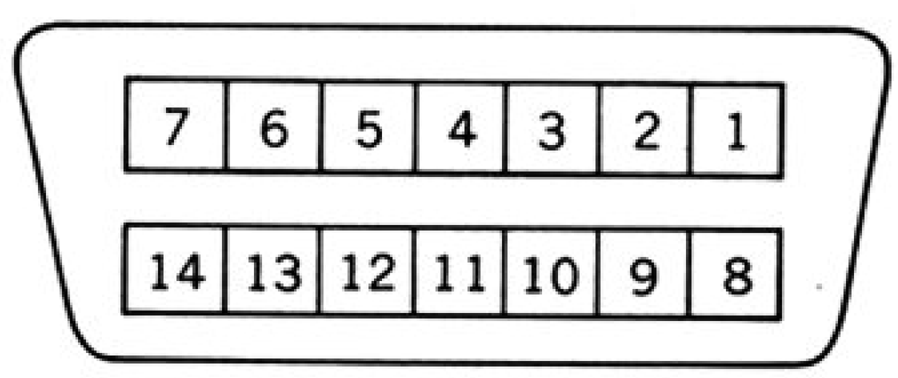


図1.11 プリンタインターフェイスの信号線

3.4 汎用入出力インターフェイス

使用IC	AY-3-8910 相当品
入出力	入力 4bit、出力 1bit、双方向 2bit (1ポートにつき)
レベル	TTL
コネクタ	AMP 9ピンコネクタ相当品 (本体側オス)

ピン番号	信号名	方　向
1	FWD	入力
2	BACK	入力
3	LEFT	入力
4	RIGHT	入力
5	+5V*	
6	TRG 1	入出力
7	TRG 2	入出力
8	COM	出力
9	GND	

ピンコネクション

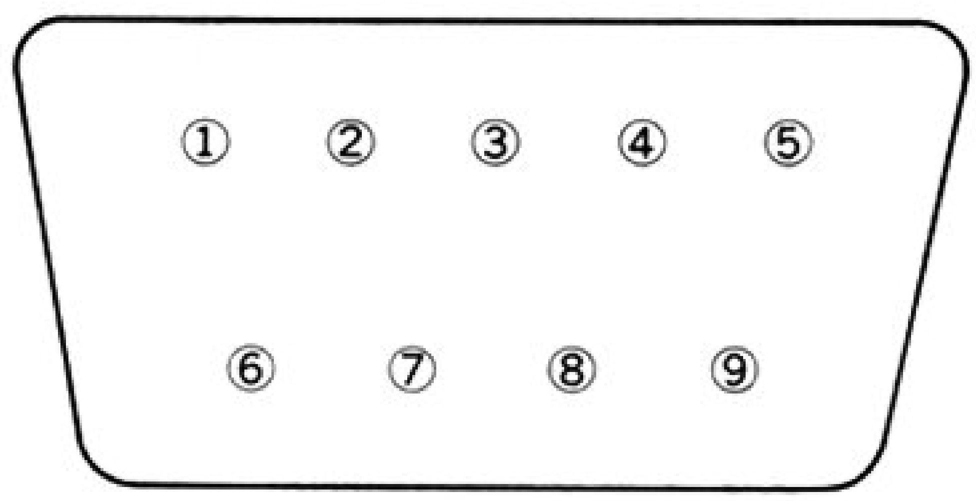


図 1.12　汎用入出力インターフェイスの信号線

注　意　*電流容量は 50mA 以上とします。

汎用入出力インターフェイス回路は、「7章　MSX2+回路例」を参照して下さい。

3.5 ジョイスティック

ジョイスティックには以下の2タイプがあります。

Aタイプ	1個のトリガボタンを持つか、複数のトリガボタンを備えていてもソフトウェアでその信号を区別することができないタイプ。
Bタイプ	2個のトリガボタンを持ち、どちらのボタンが押されているかソフトウェアで

判別できるタイプ。

市販のジョイスティックは、ほとんどがBタイプで、ソフトウェアもBタイプが前提となっています。したがって、Aタイプのジョイスティックを市販するときは、注意が必要です。

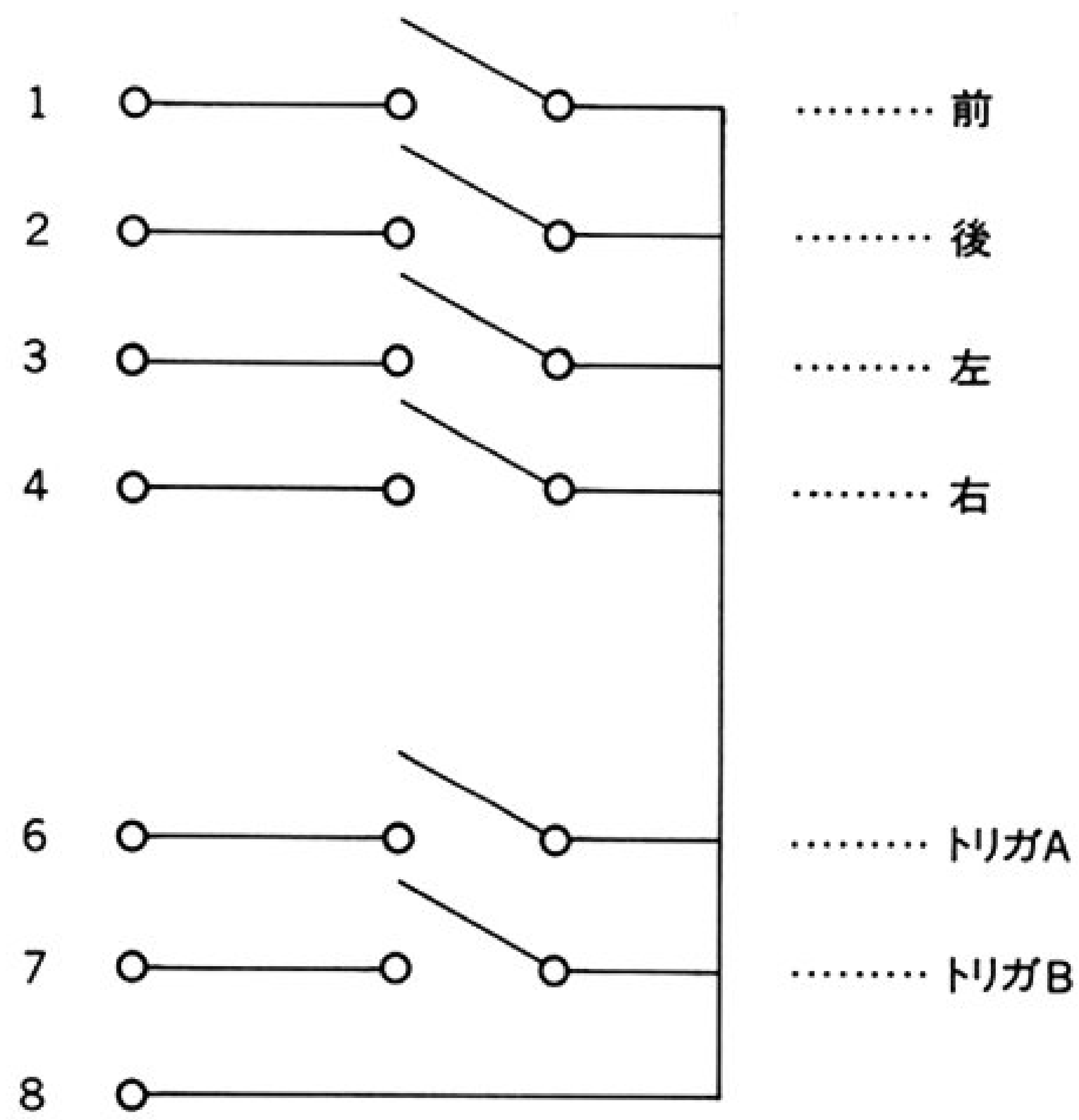


図1.13 ジョイスティック回路接続図

3.6 パドル

PDL関数がコールされると、汎用入出力ポートの8ピンにトリガパルスが送られます。このパズルがパドル内の単安定マルチバイブレータを起動します。マルチバイブレータは、パドルに接続された可変抵抗の値によって長さの異なるパルスを発生し、ポートに返してきます。

1ポートにつき、最大6個のパドルが接続可能です。

注　意

パドルはMSX turbo R以降のMSXでは、システムソフトウェアによるサポートがなくなる予定です。

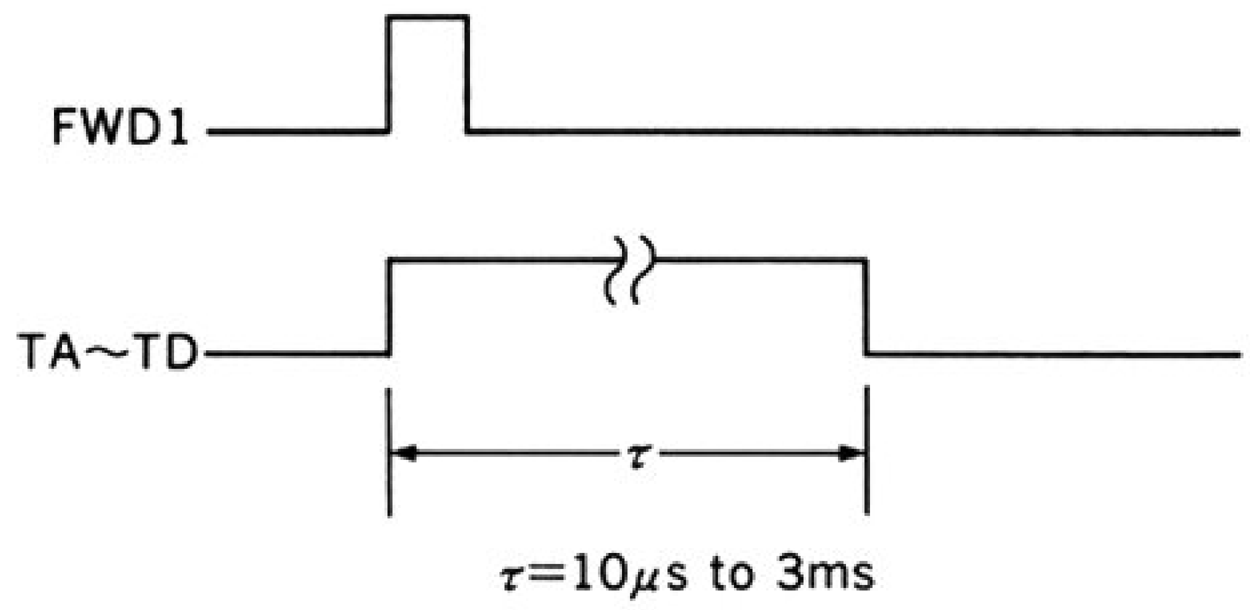


図 1.14　パドルのタイミング図

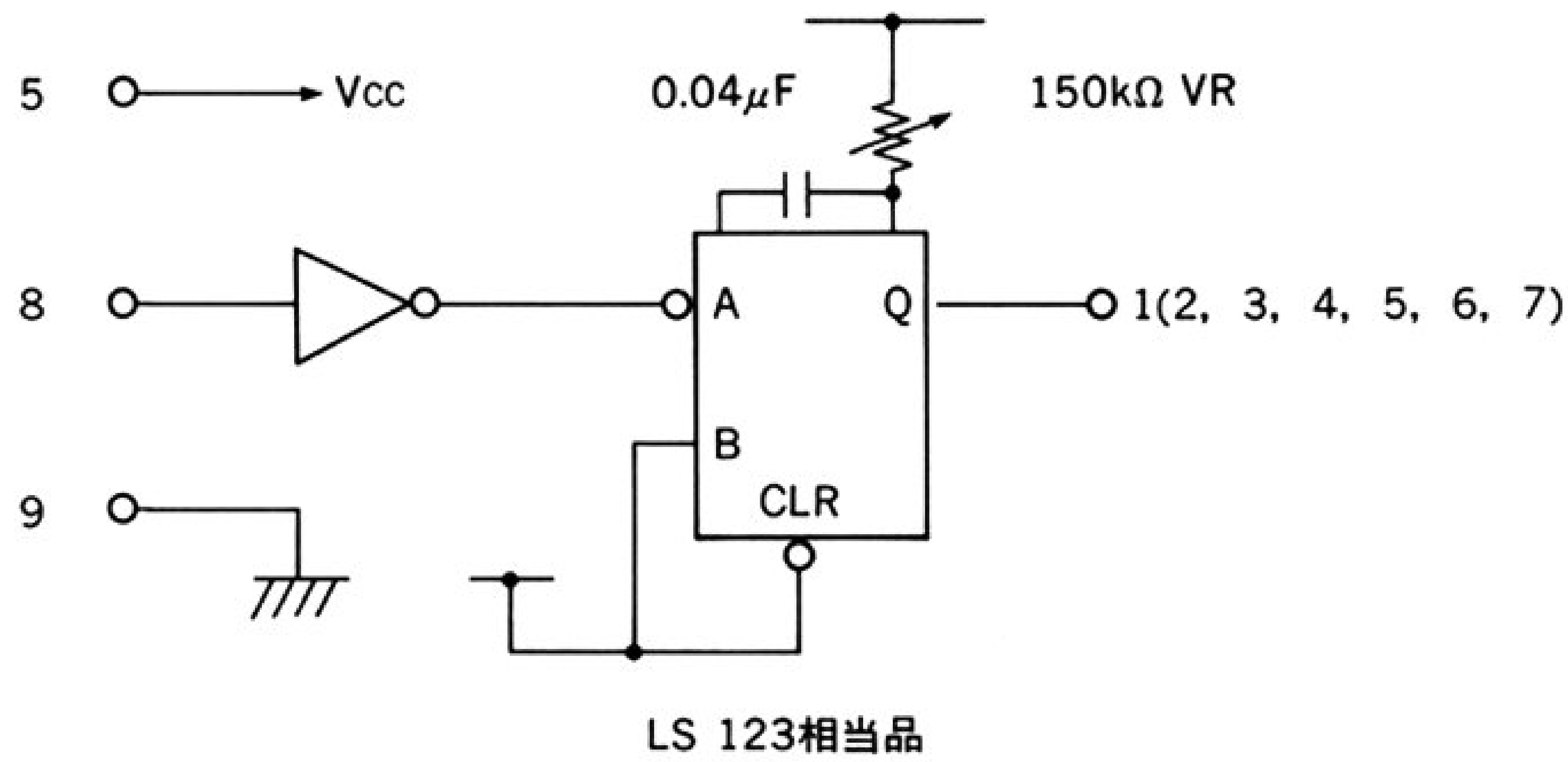


図 1.15　パドルの回路接続図

3.7 マウス

3.7.1 概要

MSX用のマウスは本体側の負担が軽く、接続が簡単であるように汎用入出力インターフェイスに接続します。また、マウスをサポートしないソフトウェアでもマウスが使用できるように、カウンタモード（マウスモード）とジョイスティックモードの2つのモードを持ちます。

表 1.3 マウスの信号線一覧

ピン番号	信号名	ジョイスティック
1	データビット1	前
2	データビット2	後
3	データビット3	左
4	データビット4	右
5	+5V	
6	左ボタン	トリガ1
7	右ボタン	トリガ2
8	STROBE	GND
9	GND	

3.7.2 カウンタモード

カウンタモードはマウス本来のモードです。マウスはX・Y方向の移動を検出し、それを8ビットの内部カウンタに保存します。そして、CPUからの要求があれば、MSX本体にデータを転送します。データ転送シーケンスは以下のとおりです。

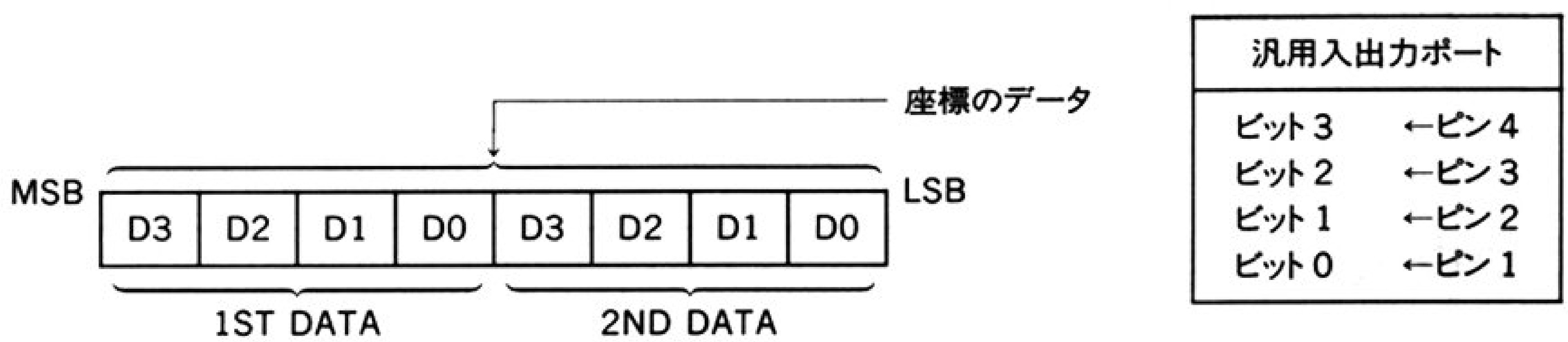


図 1.16 座標のデータ型式

マウスの座標の読み出しは、BIOSのGETPADで行います。BIOSは以下のタイミングでマウスのデータを獲得しているので、ハードウェアはこのタイミングでデータを送ります。

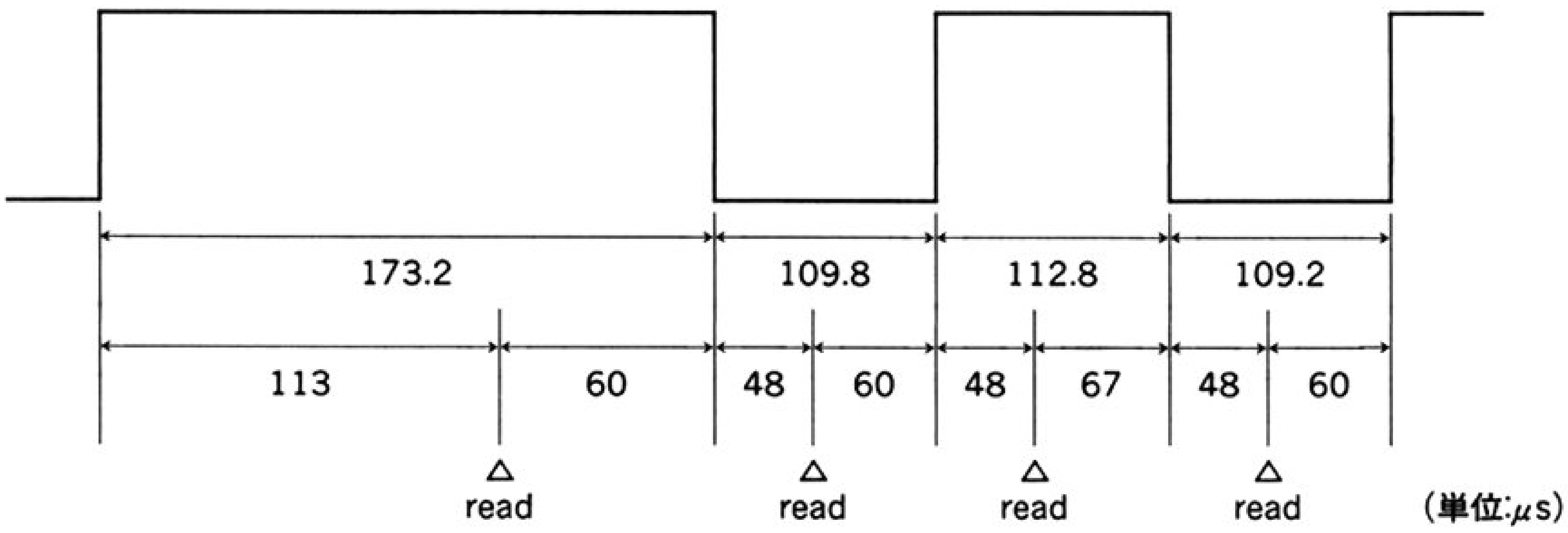


図 1.17　マウスのタイミング図

3.7.3 ジョイスティックモード

MSX のパワーON 時に、マウスの左ボタンを押していると、ジョイスティックモードになります。このモードではマウスをジョイスティックとして使用できます。

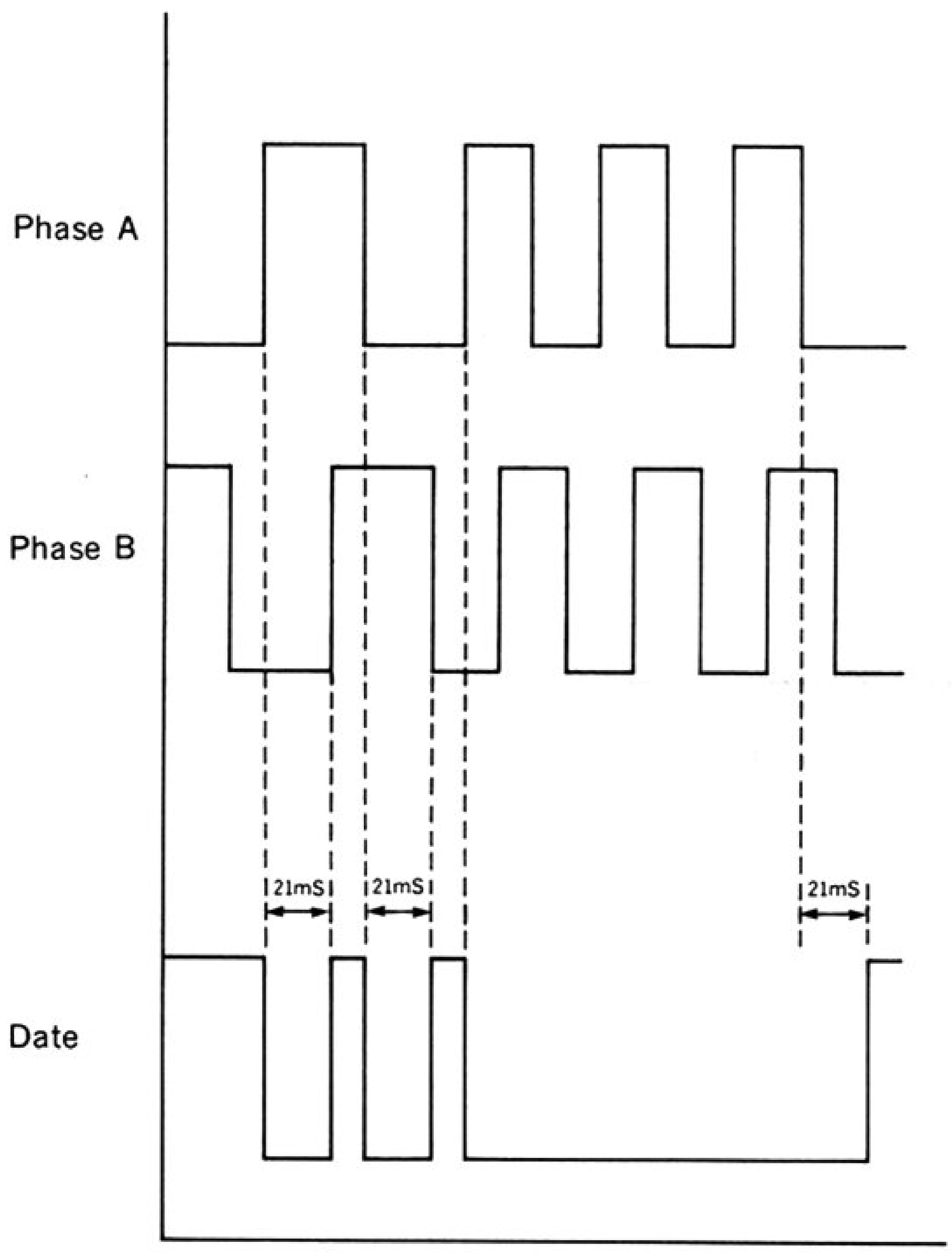


図 1.18　ジョイスティックモードのタイミング図

3.8 コネクタ

端子名	仕　様
ビデオ出力	
● RGB 信号	決まっていないが、MSX2＋では DIN 8 ピンコネクタ (DIN-45326) が一般的
●複合映像信号	DIN 5 ピンコネクタ
（ビデオ信号）	RCA 2 ピンコネクタ
● RF 変調信号	RCA 2 ピンコネクタ
カセット	DIN 8 ピンコネクタ DIN-45326
汎用入出力ポート	AMP 9 ピンコネクタ相当品
プリンタ	アンフェノール 14 ピンコネクタ相当品
カートリッジバス	2.54 ピッチ、50 ピンコネクタ
オーディオ出力	RCA 2 ピンコネクタ

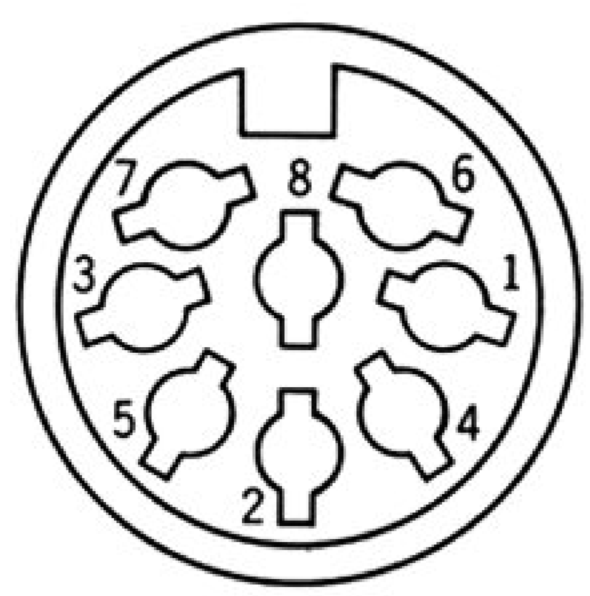
接続端子	信号名	ピンコネクション
1	GND	
2	AUDIO	
3	AV	
4	SYNC	
5	YS	
6	R	
7	G	
8	B	本体背面より見た図

図 1.19　RGB 信号 DIN 8 ピンコネクタの信号線

注　意	これは、MSX2＋、MSX2 のほとんどの機種で採用されている RGB 信号の信号線表であり、MSX の仕様として決められているわけではありません。

接続端子	信号名	ピンコネクション
1	+5V	
2	GND	
3	AUDIO	
4	MONITOR VIDEO	
5	RF VIDEO	

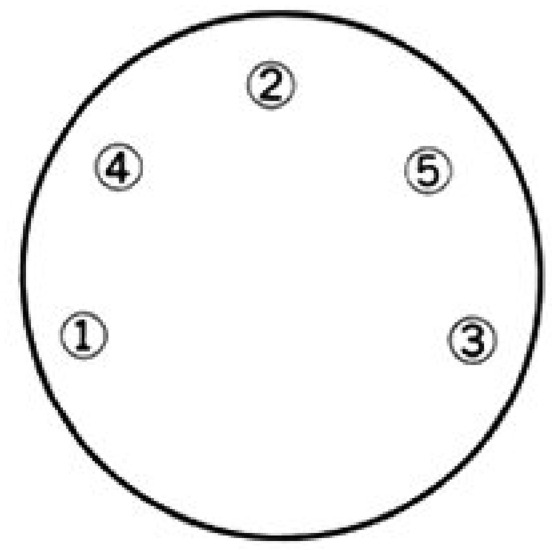


図 1.20　複合映像信号 DIN 5 ピンコネクタの信号線

注　意	これは、MSX2 の一部の機種で採用されている複合映像信号の信号線表であり、MSX の仕様として決められているわけではありません。

3.9 スロット

3.9.1 スロットの概念

この章で使われる「スロット」は、ひとつの 64K バイトのメモリ空間を構成するという点においては、「メモリバンク」という概念に近いものですが、CPU が番号を指定して選択することができるという点で、カートリッジを挿入するスロットに近いと考えられます。また、カートリッジバス上のこのスロットを選択する信号は、スロットセレクト信号と呼ぶのが最も自然であるので「スロット」と呼ぶことになりました。

この「スロット」の概念はソフトウェアの観点からとらえたものなので、実際のハードウェア (50 ピンコネクタ) を示しているわけではありません (実際のハードウェアは、「カートリッジバス」もしくは「カートリッジスロット」と呼びます)。

3.9.2 「スロット」構造をとることの利点

従来の共通バス構造で複数のメモリバンクを持つ場合には、バスにつながる各デバイスに与えられる選択信号 (スロットセレクト信号) に区別がないので、同じメモリ空間を占めるデバイスが複数個バスに接続された場合は、信号の衝突が起きてしまいシステムが正しく動作せず、場合によっては、ハードウェアの劣化をまねく恐れがあります。しかし、スロットセレクト信号によ

り、各デバイスが個別に選択される場合には、その心配はありません。さらに、同一アドレスを占めるプログラムが同時に存在できるので、システムの柔軟性・拡張性が高くなるという利点があります。

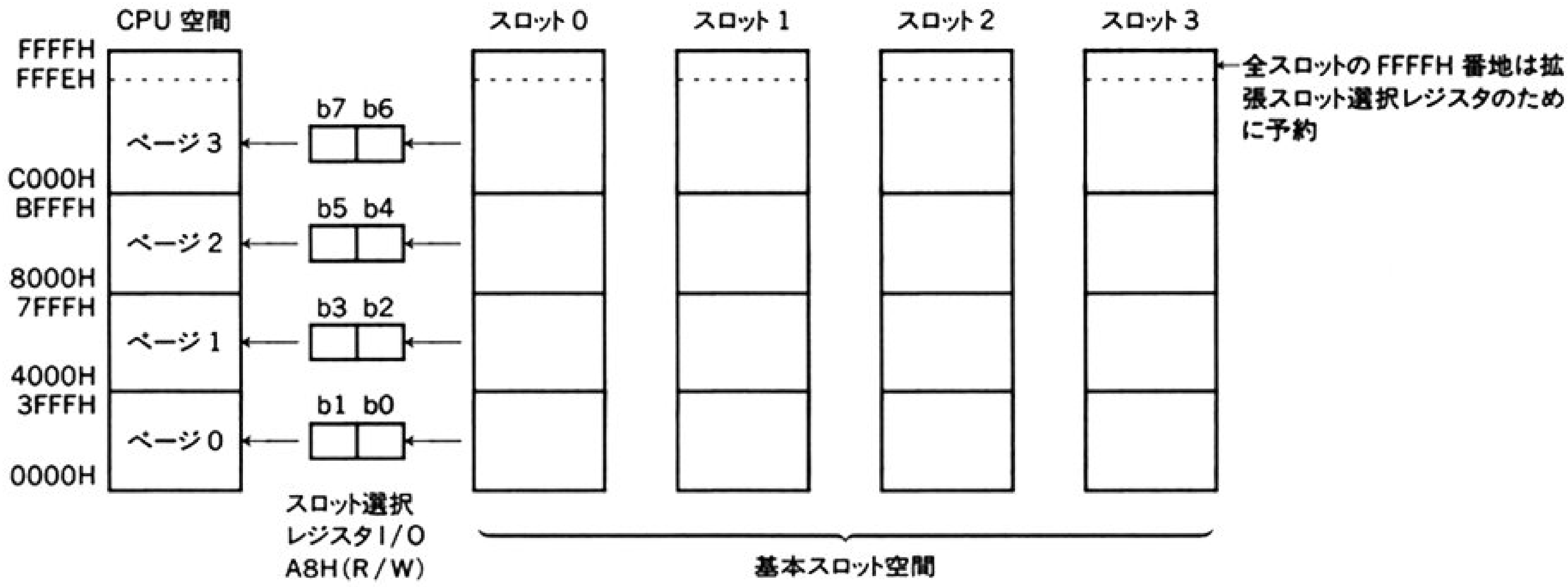


図 1.21 基本スロット

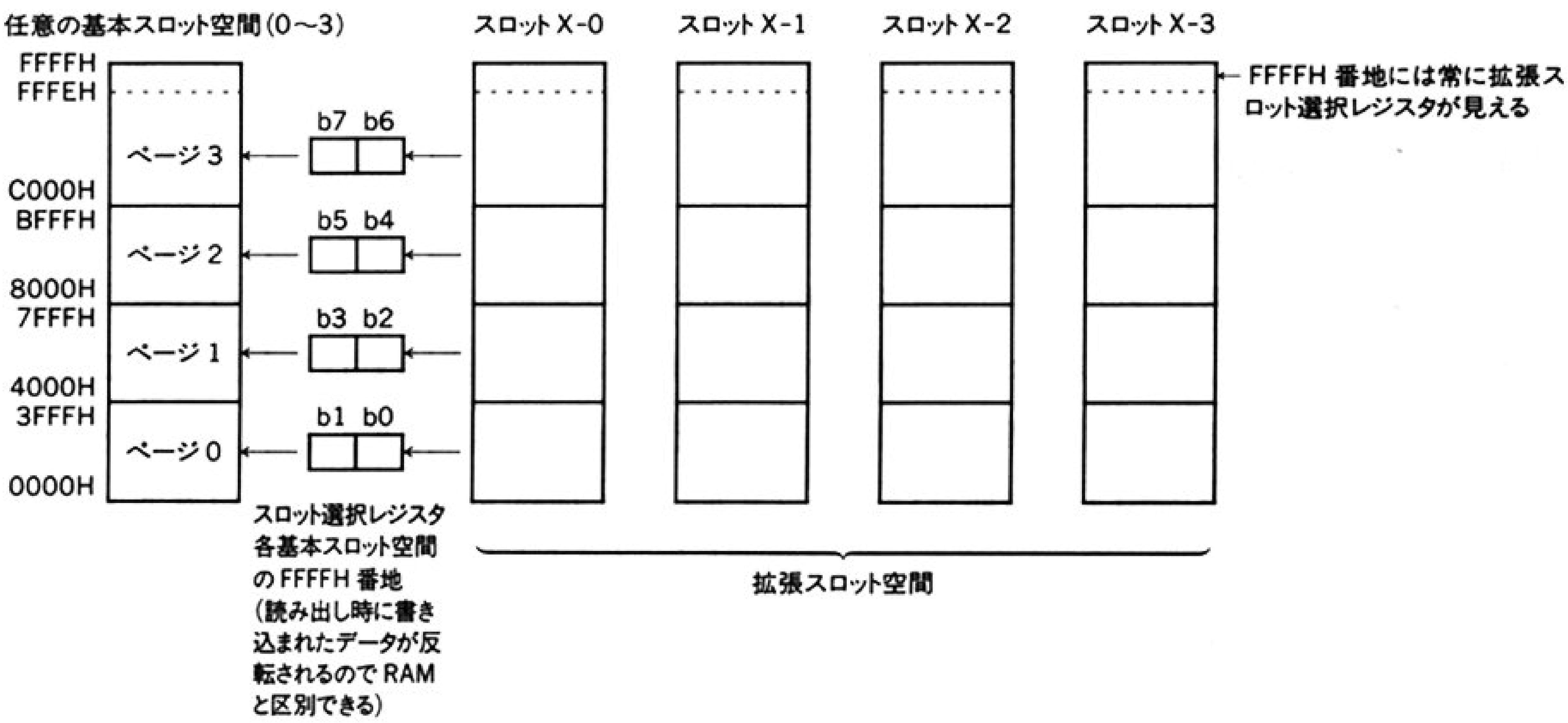


図 1.22 拡張スロット(オプション)

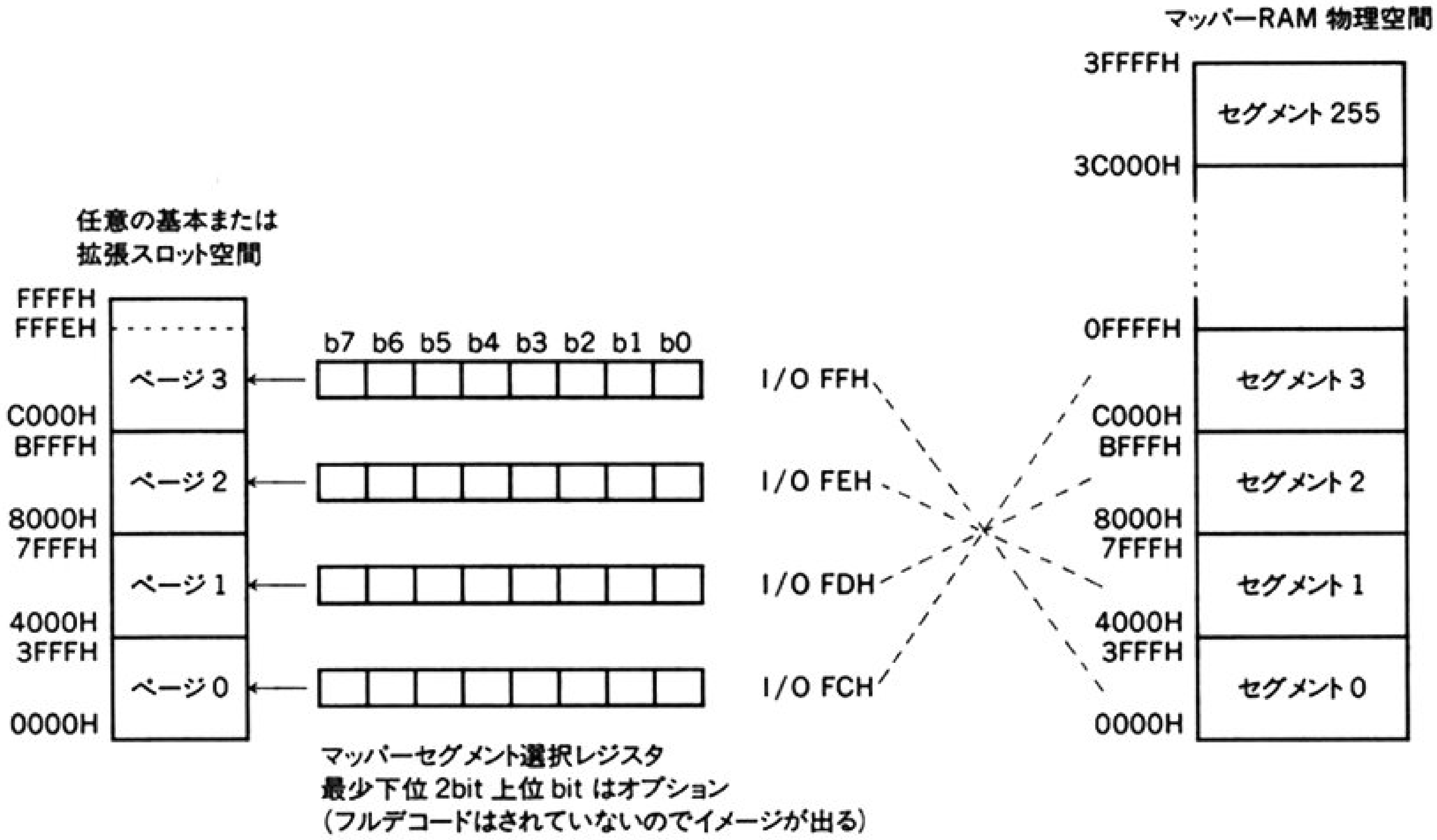


図 1.23　メモリマッパー (オプション)

3.9.3 スロットのタイミング

スロットは、以下のタイミングで動作します。

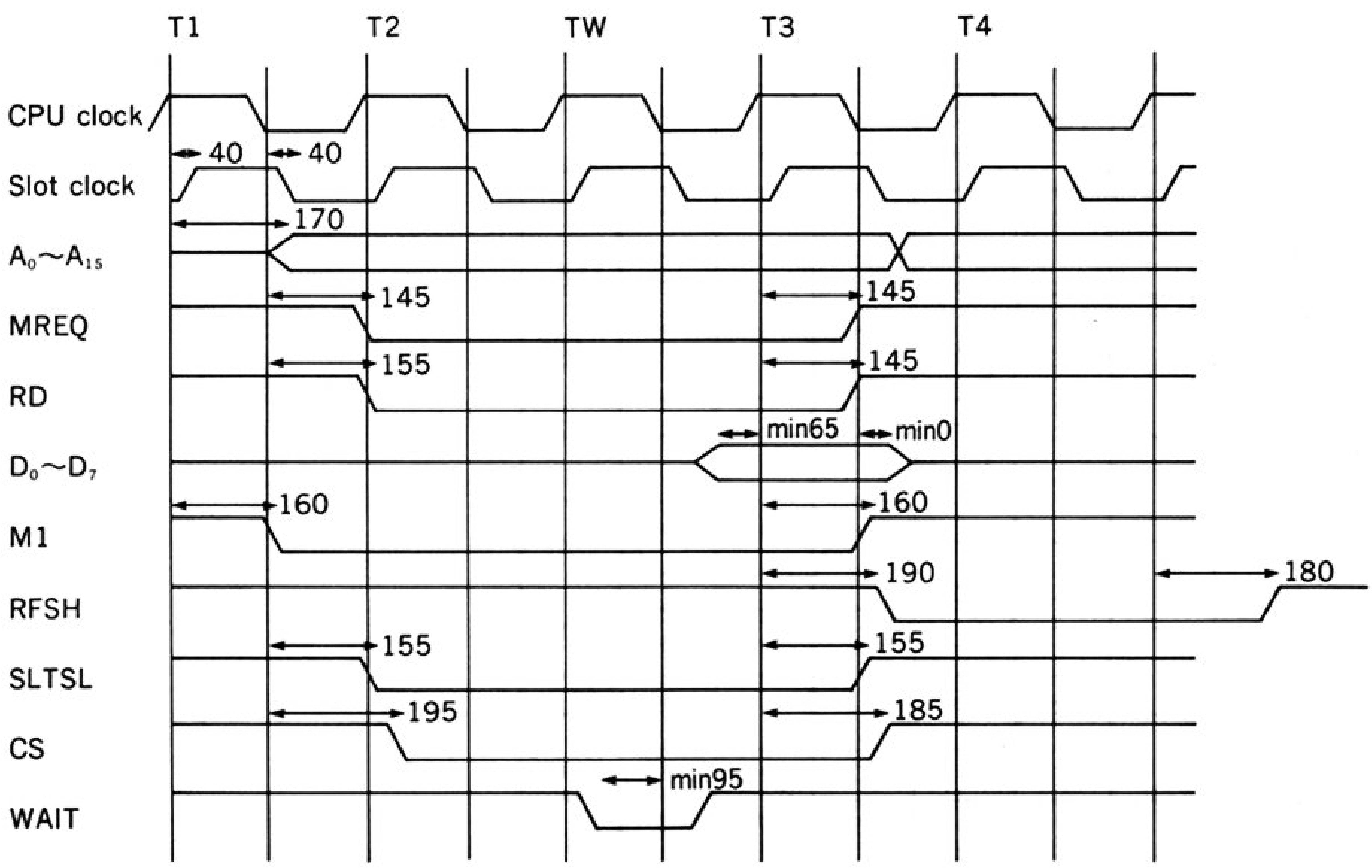


図 1.24　M1 サイクル (基本スロット)

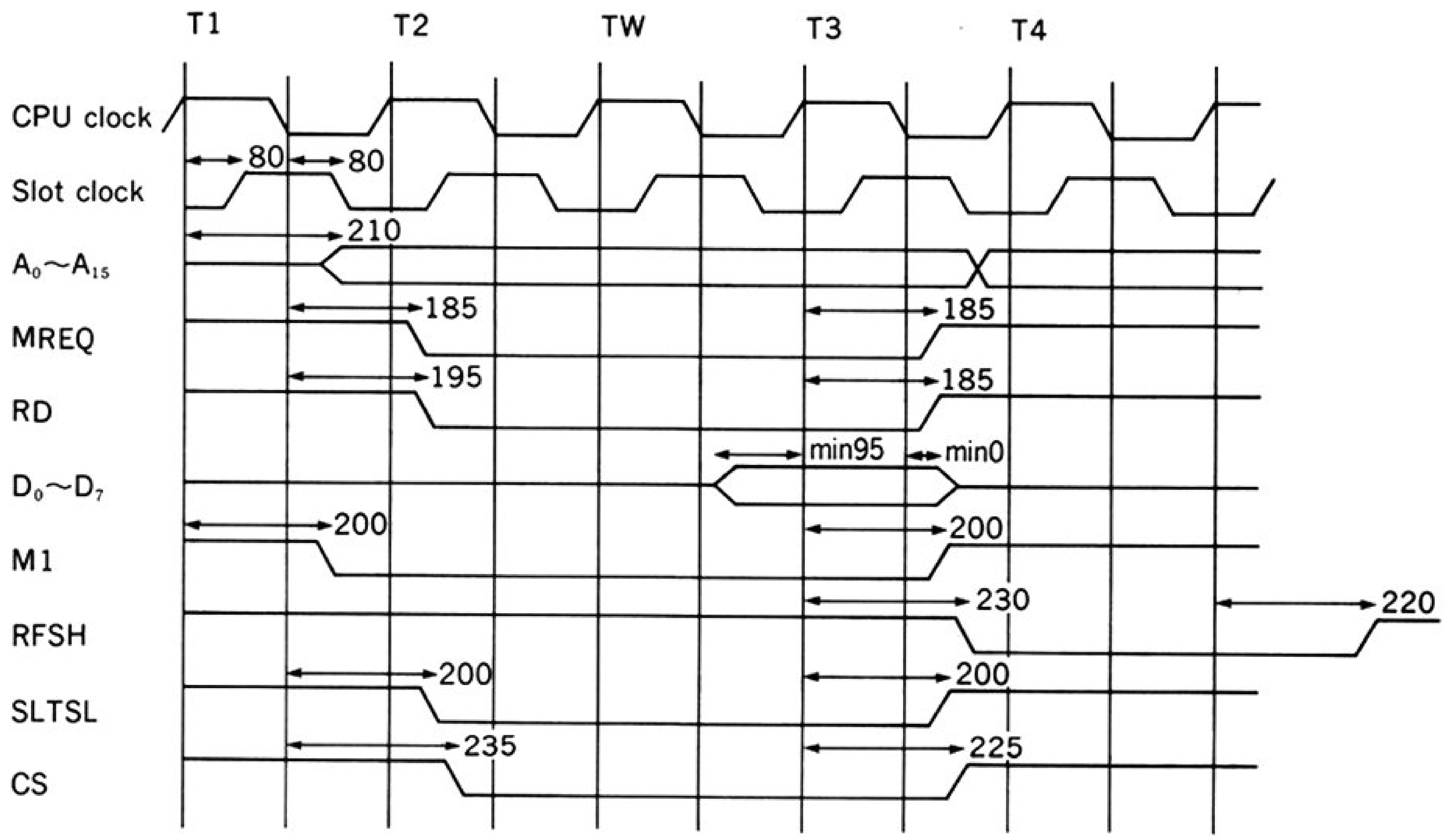


図 1.25 M1 サイクル（拡張スロット）

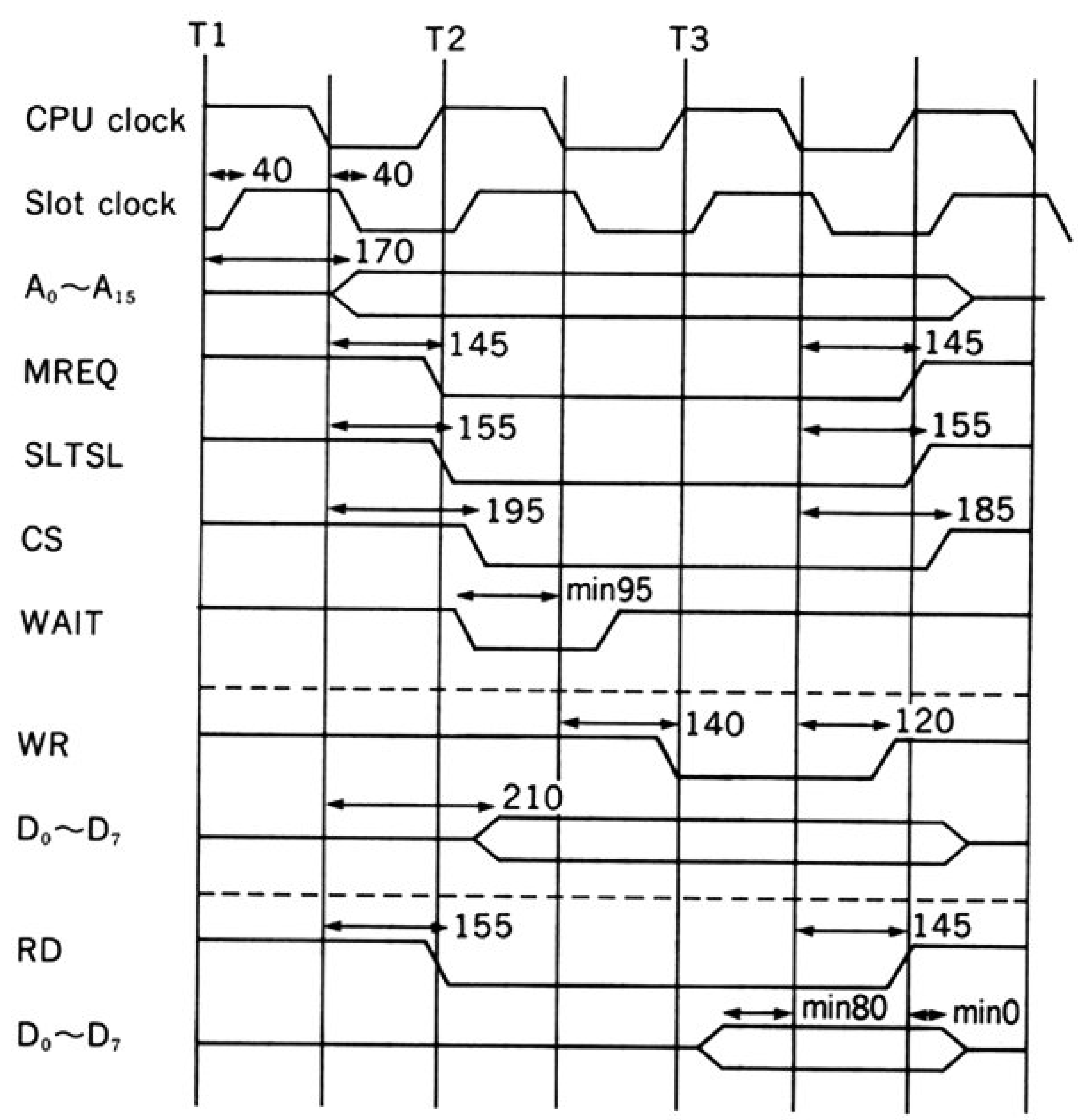


図 1.26 メモリサイクル（基本スロット）

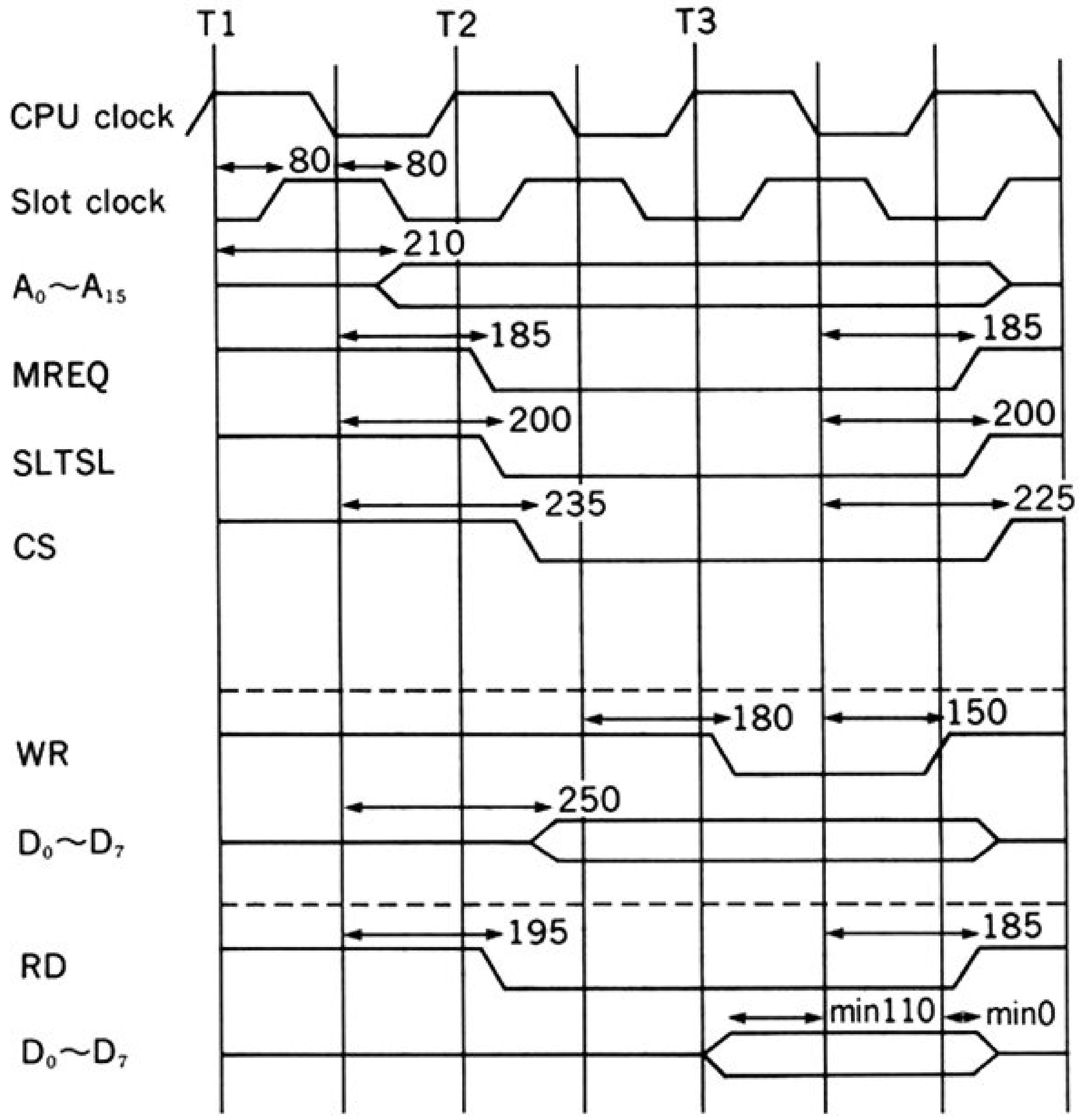


図 1.27 メモリサイクル（拡張スロット）

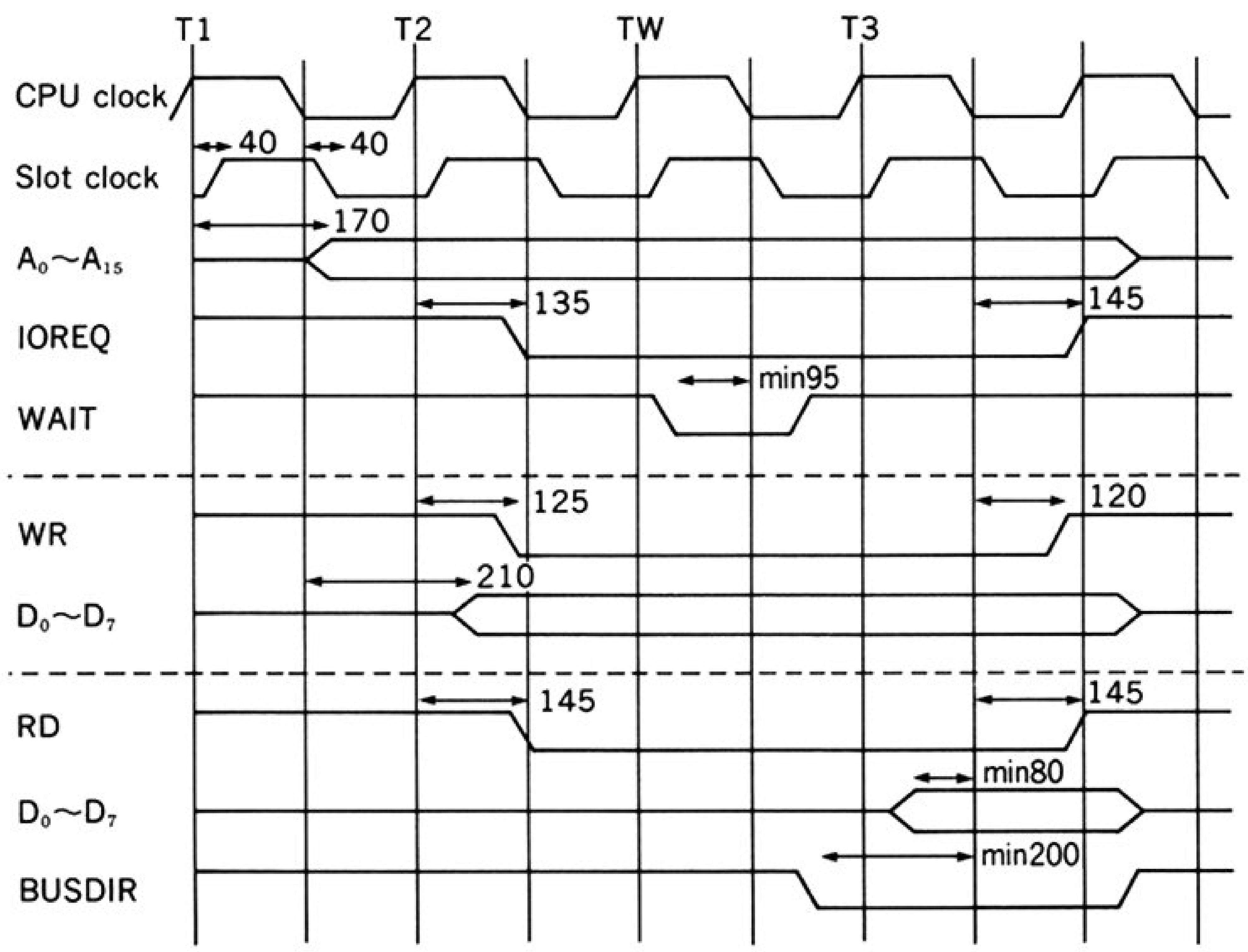


図 1.28 I/O サイクル（基本スロット）

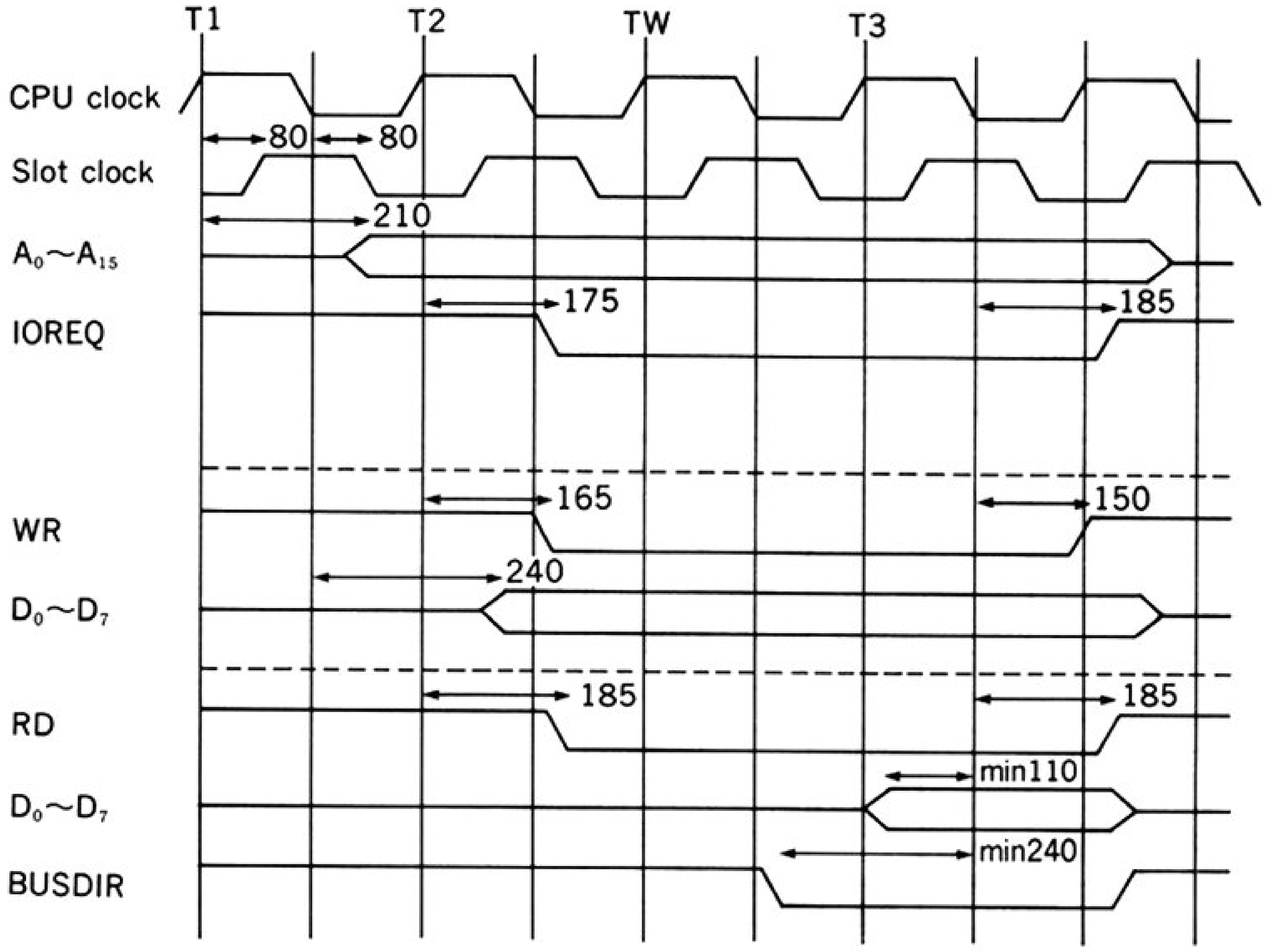


図 1.29　I/O サイクル（拡張スロット）

4章 カートリッジ

4.1 カートリッジ仕様

MSX のカートリッジは次のような仕様になっています。

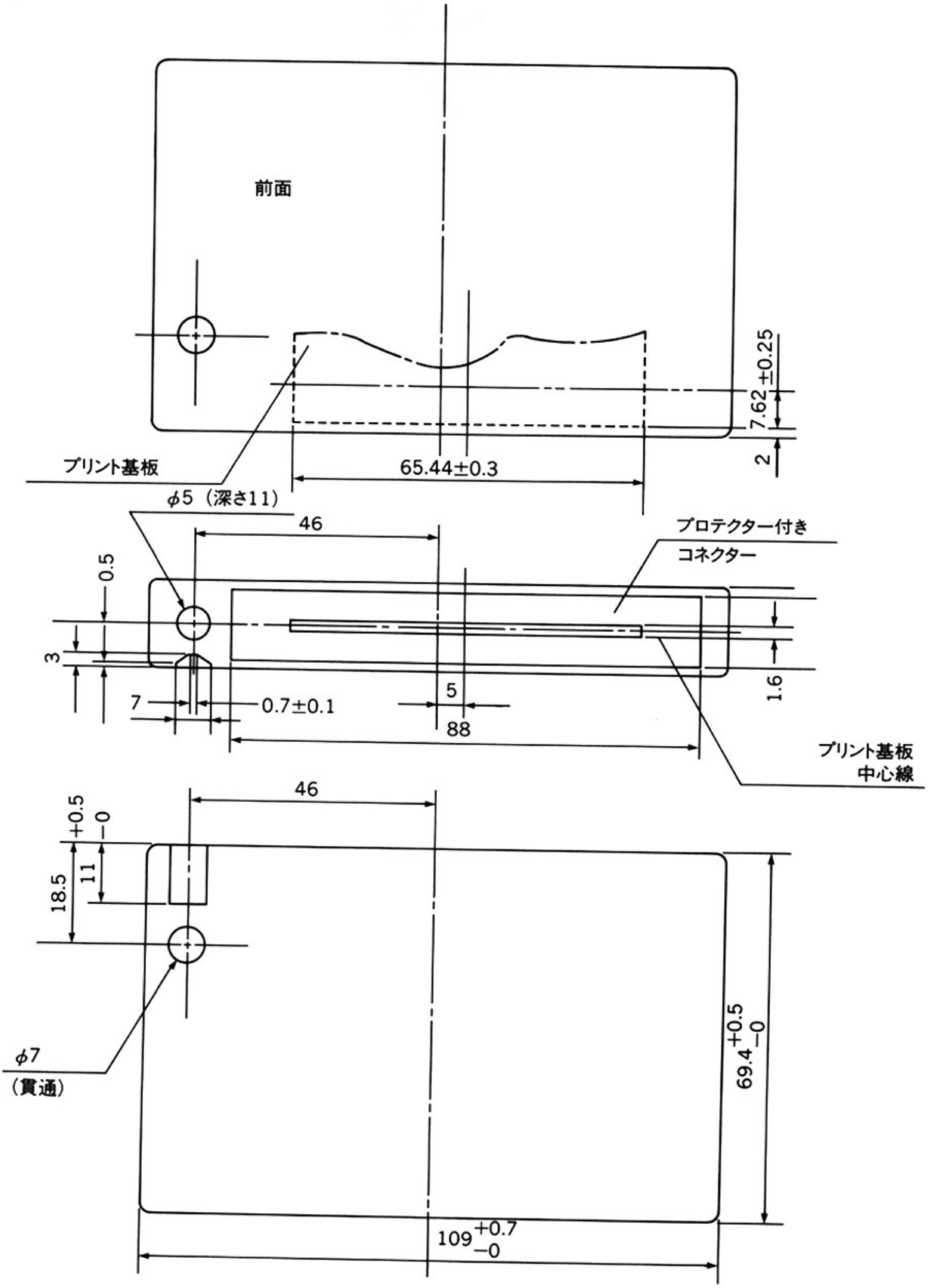


図1.30 MSXカートリッジ標準仕様

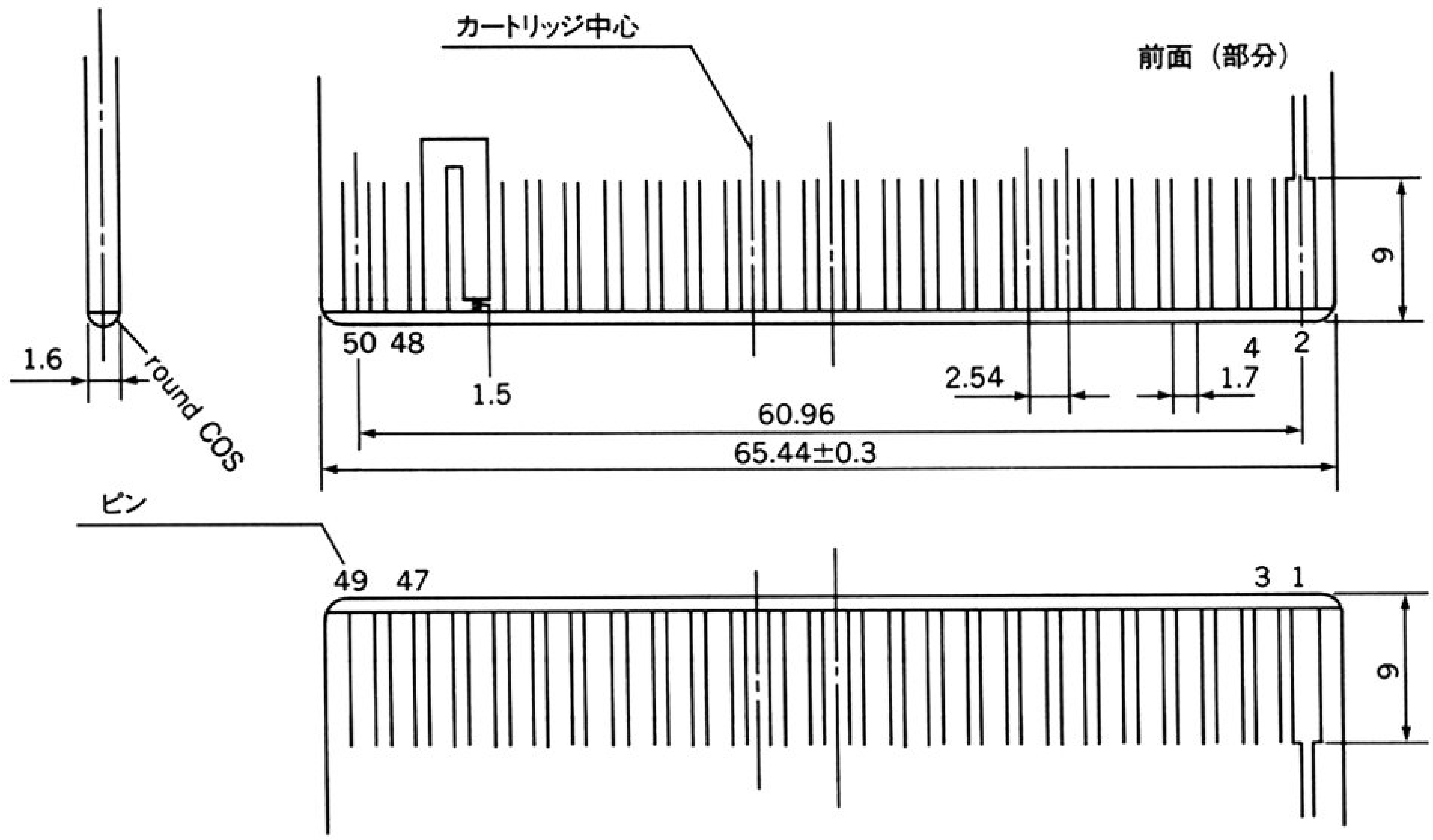


図 1.31 プリント基板のパターン

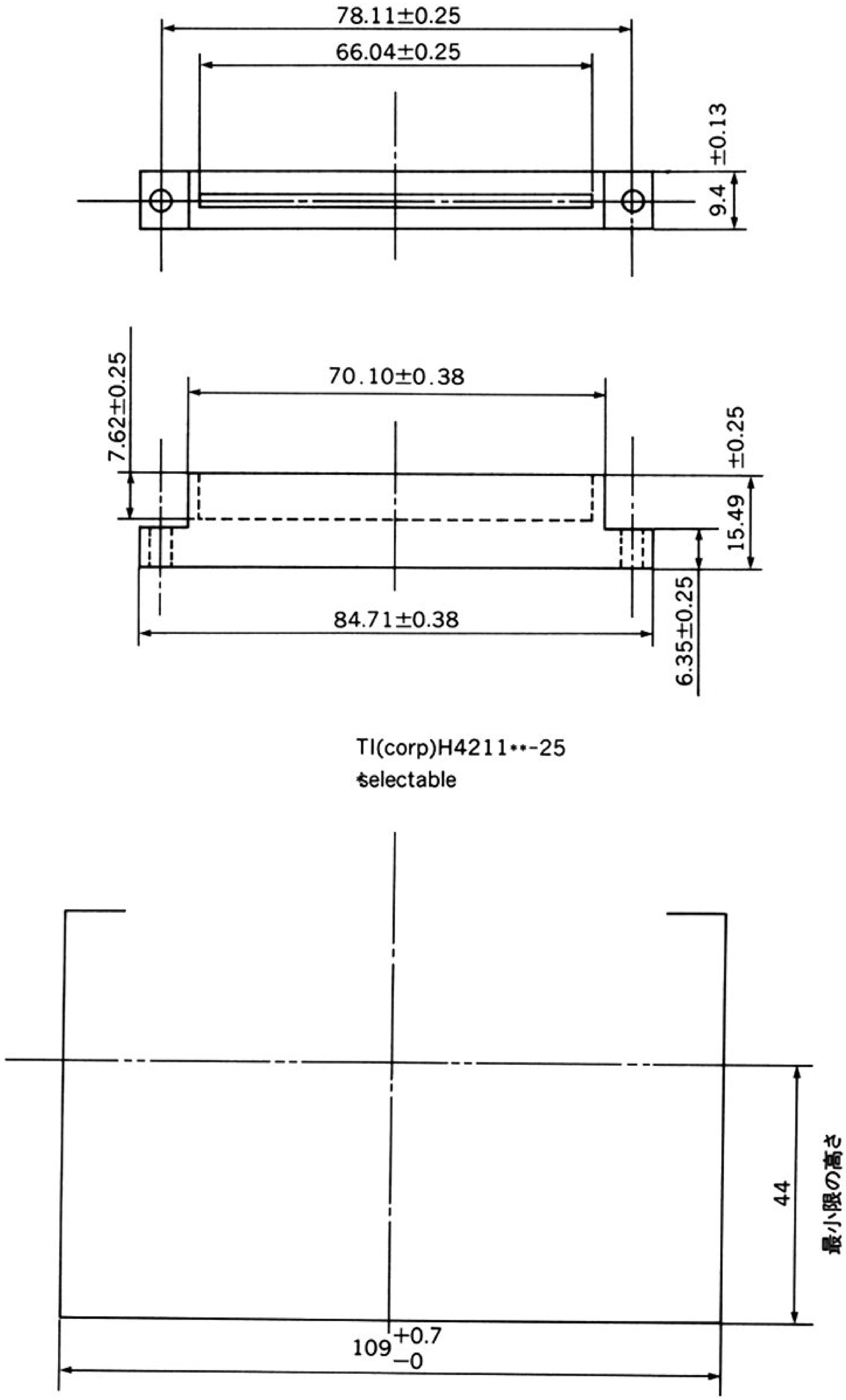


図 1.32 コネクタ例

注　意

カートリッジのコネクタ側の端から、44mm 以内の部分に突起（スイッチ、コネクタなど）をつけてはいけません。

4.2 カートリッジバス

4.2.1 信号一覧

表 1.4　カートリッジバスの信号線一覧

ピン番号	名　称	入出力*1	ピン番号	名　称	入出力*1
1	$\overline{\text{CS1}}$	O	2	$\overline{\text{CS2}}$	O
3	$\overline{\text{CS12}}$	O	4	$\overline{\text{SLTSL}}$	O
5	予約*2	−	6	$\overline{\text{RFSH}}$	O
7	$\overline{\text{WAIT}}$*3	I	8	$\overline{\text{INT}}$*3	I
9	$\overline{\text{M1}}$	O	10	$\overline{\text{BUSDIR}}$	I
11	$\overline{\text{IORQ}}$	O	12	$\overline{\text{MERQ}}$	O
13	$\overline{\text{WR}}$	O	14	$\overline{\text{RD}}$	O
15	$\overline{\text{RESET}}$	O	16	予約*2	−
17	A9	O	18	A15	O
19	A11	O	20	A10	O
21	A7	O	22	A6	O
23	A12	O	24	A8	O
25	A14	O	26	A13	O
27	A1	O	28	A0	O
29	A3	O	30	A2	O
31	A5	O	32	A4	O
23	D1	I/O	34	D0	I/O
23	D3	I/O	34	D2	I/O
23	D5	I/O	34	D4	I/O
23	D7	I/O	34	D6	I/O
41	GND	−	42	CLOCK	O
43	GND	−	44	SW1	−
45	+5V	−	46	SW2	−
47	+5V	−	48	+12V	−
49	SUNDIN	I	50	−12V	−

注　意

*1 本体を基準にした入出力の区別
*2 予約は使用禁止端子
*3 オープンコレクタ出力

4.2.2 信号線の説明

表 1.5　カートリッジバス信号の説明一覧

ピン番号	名　称	内　容
1	$\overline{\mathrm{CS1}}$	ROM 4000H～7FFFH 番地セレクト信号
2	$\overline{\mathrm{CS2}}$	ROM 8000H～BFFFH 番地セレクト信号
3	$\overline{\mathrm{CS12}}$	ROM 4000H～BFFFH 番地セレクト信号（256K ROM 用）
4	$\overline{\mathrm{SLTSL}}$	スロットセレクト信号
		各スロット毎のそのスロット固有のセレクト信号
5	予約	将来のための予約（使用禁止）
6	$\overline{\mathrm{RFSH}}$	リフレッシュサイクル信号
7	$\overline{\mathrm{WAIT}}$	CPU への WAIT 要求信号
8	$\overline{\mathrm{INT}}$	CPU への割り込み要求信号
9	$\overline{\mathrm{M1}}$	CPU のフェッチサイクルを表す信号
10	$\overline{\mathrm{BUSDIR}}$	外部データバスバッファの方向を制御する信号
		カートリッジがセレクトされ、データを送出するタイミングでメモリを除く各カートリッジより L レベルを出力する。
		(I/O リードとアドレスをデコードした信号を OR した信号を入れる。詳細は 7 章を参照)
11	$\overline{\mathrm{IORQ}}$	I/O リクエスト信号
12	$\overline{\mathrm{MERQ}}$	メモリリクエスト信号
13	$\overline{\mathrm{WR}}$	ライトタイミング信号
14	$\overline{\mathrm{RD}}$	リードタイミング信号
15	$\overline{\mathrm{RESET}}$	システムリセット信号
16	予約	将来のための予約（使用禁止）
17～32	A0～A15	アドレスバス信号
33～40	D0～D7	データバス信号
41	GND	電源および信号グランド
42	CLOCK	CPU クロック　3.579545MHz
43	GND	信号グランド
44、46	SW1、SW2	抜き差しプロテクト用
45、47	+5V	+5V 電源
48	+12V	+12V 電源
49	SUNDIN	サウンド入力信号（−5dBm）
50	−12V	−12V 電源

4.2.3 カートリッジバス接続条件

1. ファンイン、ファンアウト（LS TTL）

○ファンイン，ファンアウト(LS TTL)
データ・アドレスバス

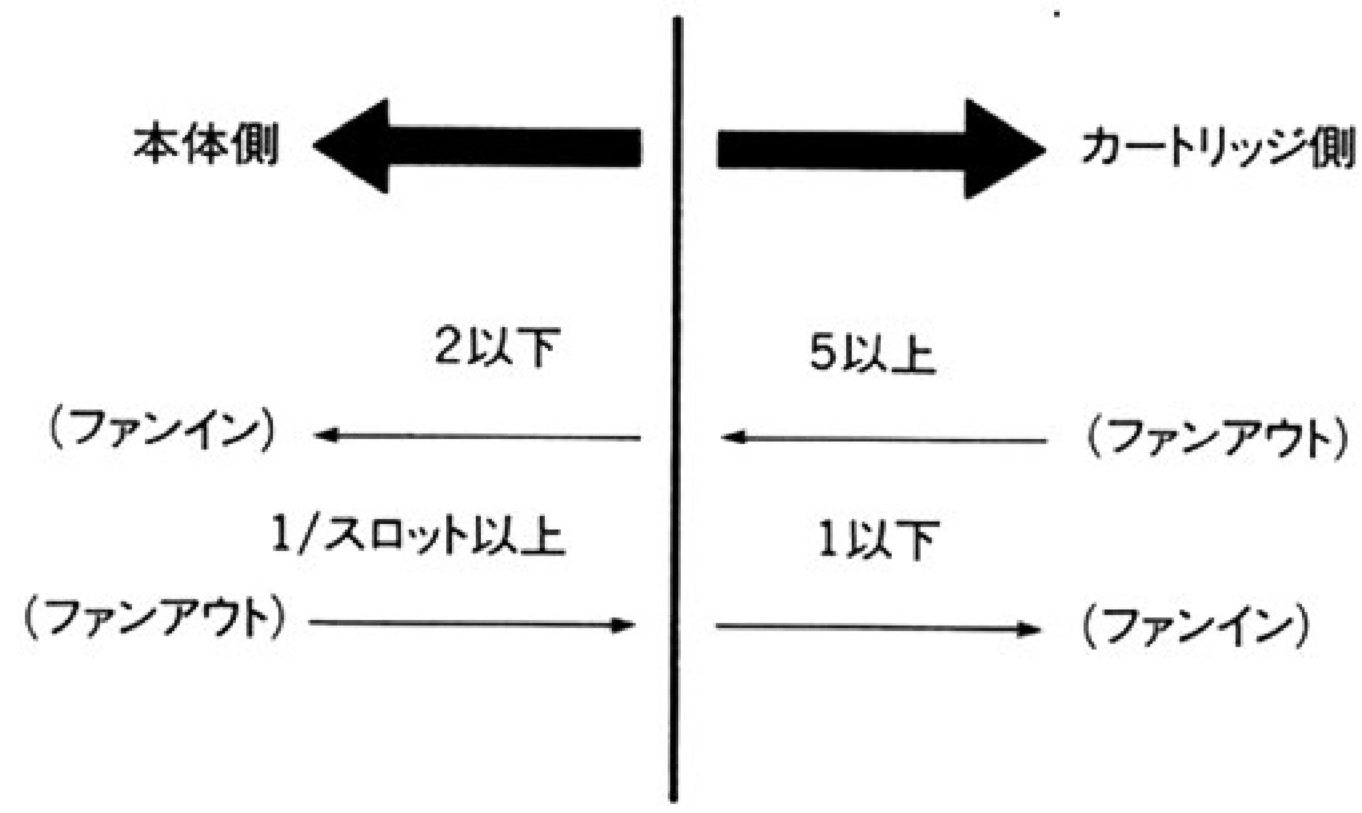


図 1.33　データ・アドレスバスのファンイン、ファンアウト

○コントロール信号

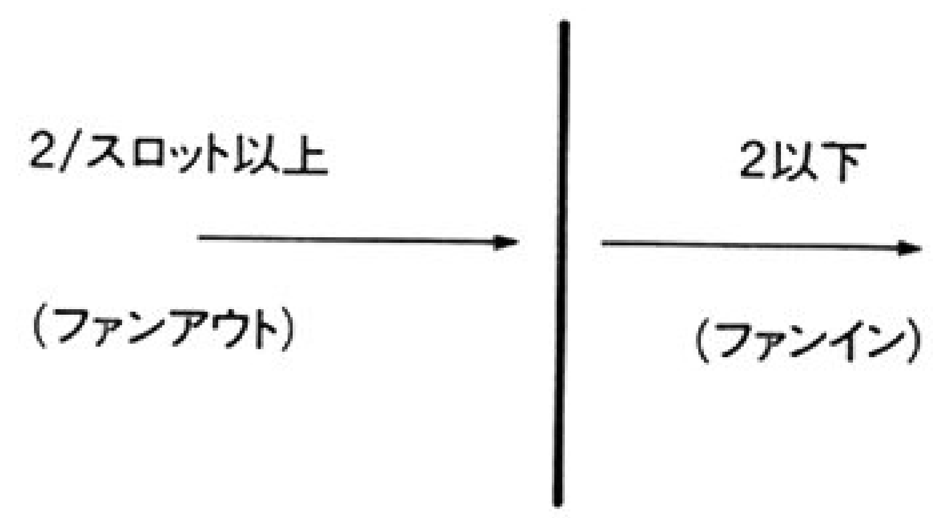


図 1.34　コントロール信号のファンイン、ファンアウト

2. 電圧レベル

TTL レベル

4.3 電源容量

+5V　　300mA/各スロット
+12V　　50mA/スロット合計
−12V　　50mA/スロット合計

4.4 拡張スロットセレクト信号

拡張スロットセレクト信号の回路は、「7章　MSX2+回路例」を参照して下さい。

5章 システムを拡張する際の注意

5.1 スロットの拡張

基本構成の 4 スロットを基本スロットと呼びます。基本スロット以上のスロットをシステムに追加する場合には、基本スロットから拡張します。すなわち、拡張されたスロットを選択するには、まず拡張スロットが接続されている基本スロットを選択し、さらに、その中で追加されている拡張スロットを選択する構造を取るものとします。

この拡張されたカートリッジバスの双方向データバスバッファの向きを制御するのが BUSDIR 信号です。

メモリ(RAM や ROM など)だけのカートリッジ(ゲームカートリッジなど)は、この BUSDIR 信号を操作する必要がないので、カートリッジのコストを低くすることができます。しかし、CPU に対し信号を送る I/O デバイスを持ったカートリッジは、CPU にデータを送るタイミングに合わせて BUSDIR 信号を L レベルにして、バッファの向きを CPU 側に変えなくてはなりません。

拡張スロットのスロットセレクト信号を作成するためのスロットセレクトレジスタは、拡張スロットのメモリアドレス FFFFH 番地に置きます。そして、この拡張スロットセレクトレジスタを RAM と区別するため、このレジスタを読み出すとレジスタの内容のコンプリメントが読み出されるものとします。こうすることにより、基本スロットの 1 から 3 のどのスロットからでも、また、2 つ以上のスロットからでも拡張スロットを設けることができます。

5.2 I/O の拡張

MSX システムに予め規定されていない I/O 装置を追加する場合は、その I/O 装置のアドレスをどこに割り当てるかが問題となります。I/O アドレスの 40H 番地から FFH 番地までは、

MSX システムの標準装置のための領域なので、その装置が標準装置として認められない限り使うことはできません。00H 番地から 3FH 番地は規定されていないので使用することはできますが、複数の装置が競合を起こし、システムが動作しない事態を引き起こす可能性があります。

これを避けるために、原則としてI/O装置もメモリ空間に割り当てることにします。

6章 アドレスマップ

6.1 メモリマップ例

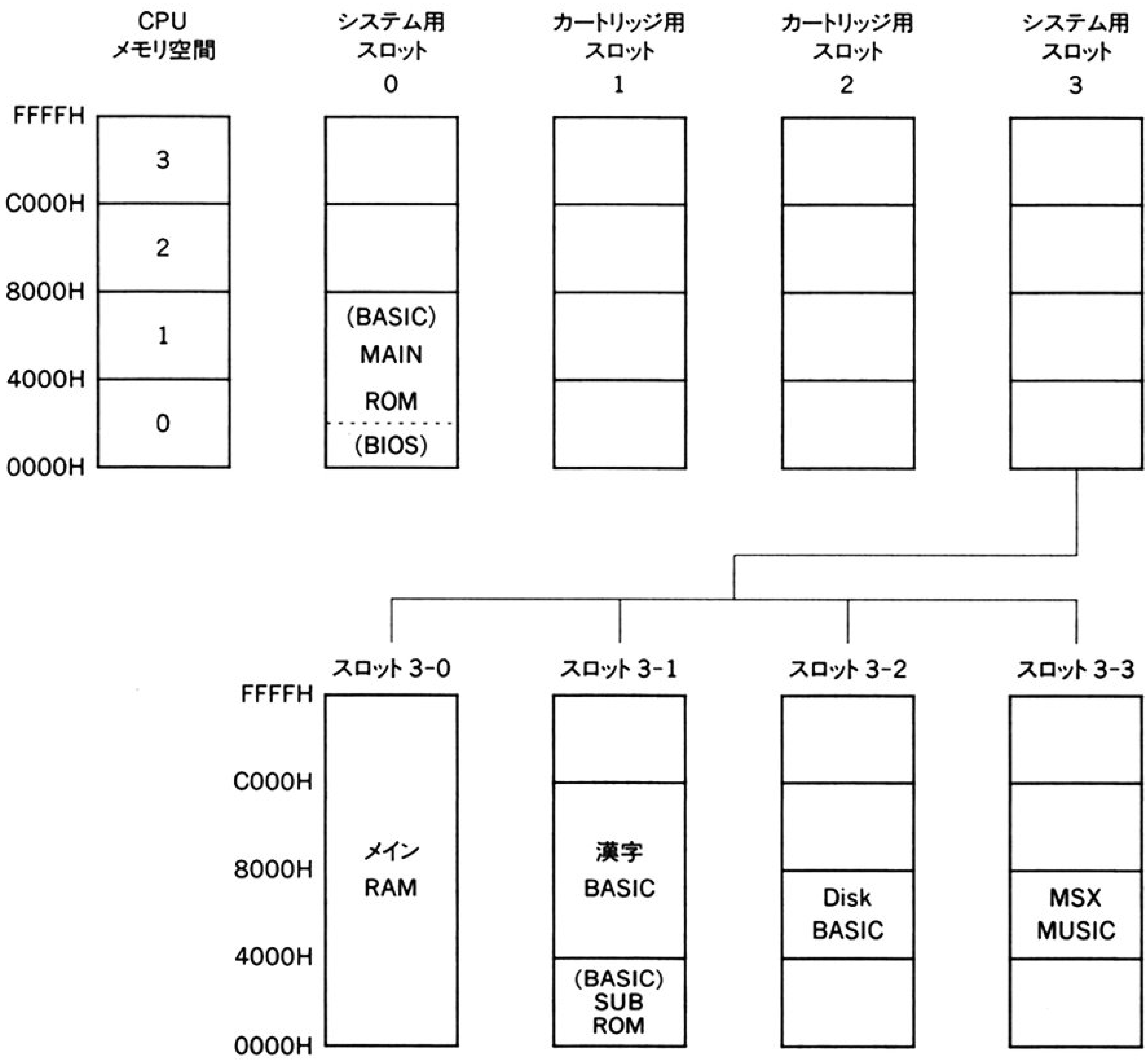
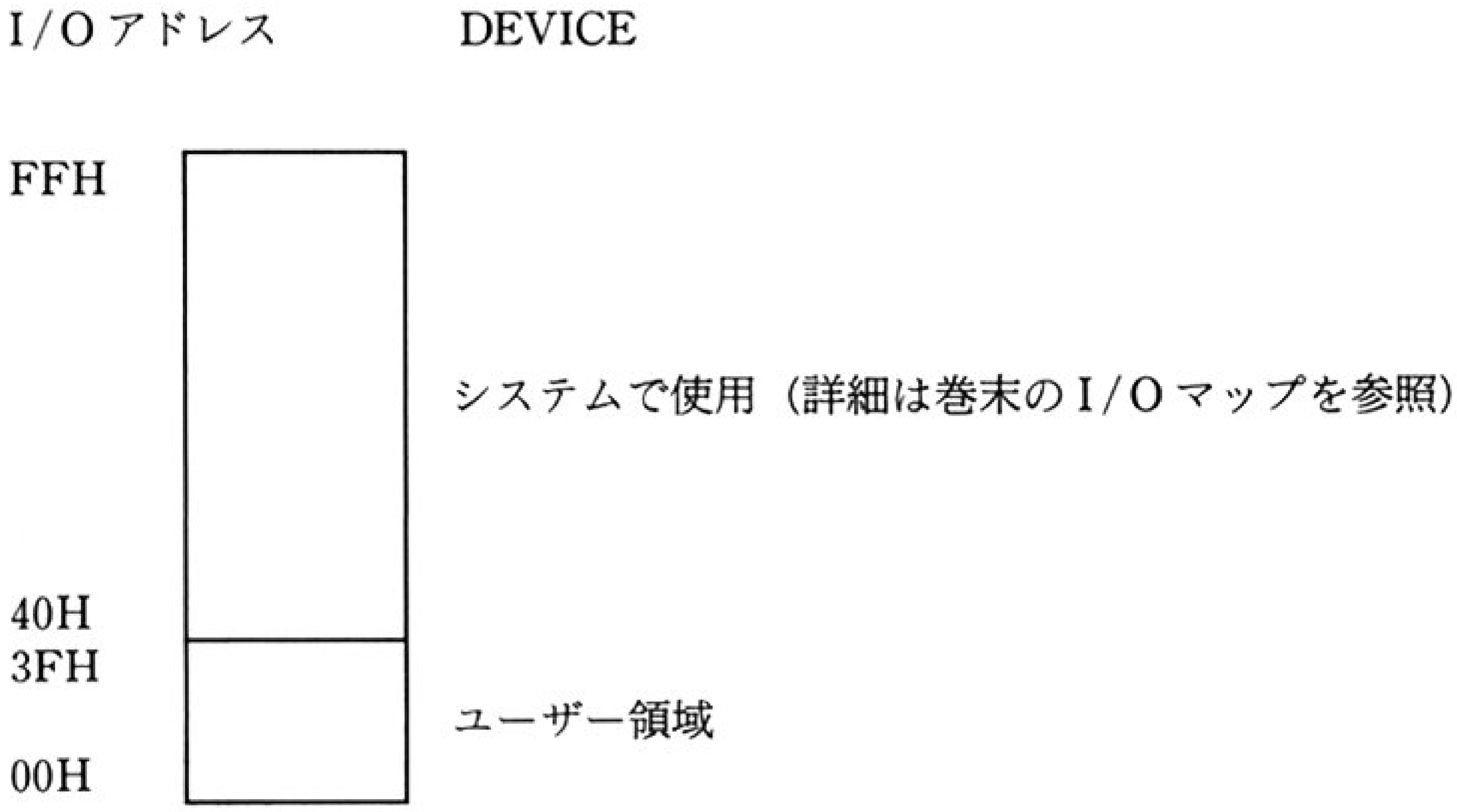


図 1.35　メモリマップ例

6.2 I/O マップ



6.3 I/O アドレスの割り当てについての注意

I/Oアドレスの 40H〜FFH はシステム用として使用します。この中で、現在使用していないアドレスは、将来のシステム拡張用として予約されています。詳細は巻末の「I/Oマップ」を参照してください。

I/Oポートを直接操作するようなソフトウェアの作成は、ソフトウェアの互換性を損なう可能性が高くなります。そのため、ハードウェアに依存しないソフトウェアを作成するために、I/Oデバイスは常に BIOS を通してアクセスするようにしてください。

ただし、VDP のアクセスに関しては、高速な画面操作を行うために I/O ポートを直接アクセスしてかまいません。しかし、その場合も、MAIN ROM の 6、7 番地を調べて、ポートアドレスを確認してください。詳しくは、「第4部　VDP」を参照して下さい。

I/Oポートの 00H〜3FH はユーザーに解放されていますが、同一アドレスに複数の周辺装置が実装されることを防ぐために、できるだけメモリマップドI/Oにしてください。

フロッピーディスクコントローラ (FDC) はI/Oエリアに実装することもできますが、その場合には回路は外部からの信号によって遮断される機能を持たなければなりません。そして、FDC がアクセスされるときに限って I/O に接続されます。これによって、システムが複数の異なるフロッピーディスクインターフェイスを扱うことが可能になります。

市販のI/O装置のために、どうしてもI/Oアドレスが必要になった場合は、株式会社アスキーシステム事業部で割り当てを行います。I/O装置の製造前に必ず連絡して、I/Oアドレスの割り当てを受けて下さい。

6.4 PPIビット割り当て

表1.6 PPIビット割り当て一覧

ポート	ビット	I/O	信号名	内 容
A	0 1	出力	CS0L CS0H	0000H～3FFFH番地の スロット指定信号
	2 3		CS1L CS1H	4000H～7FFFH番地の スロット指定信号
	4 5		CS2L CS2H	8000H～BFFFH番地の スロット指定信号
	6 7		CS3L CS3H	C000H～FFFFH番地の スロット指定番号
B	0 〜 7	入力		キーボードリターン信号
C	0 1 2 3	出力	KB0 KB1 KB2 KB3	キーボードスキャン信号
	4		CASON	カセットコントロール (L-ON) (カセットレコーダーのON・OFF)
	5		CASW	カセット書き込み信号
	6		CAPS	CAPSランプ信号 (Lで点灯)
	7		SOUND	ソフトによるサウンド出力 (キーボードのクリック音)

6.5 PSGビット割り当て

表 1.7 PSG ビット割り当て

ポート	ビット	I/O	接続コネクタ(ピン番号)	ジョイスティック使用時の信号
A	0	入力	J3-1 ピン *1 J4-1 ピン *2	FWD1 FWD2
	1		J3-2 ピン *1 J4-2 ピン *2	BACK1 BACK2
	2		J3-3 ピン *1 J4-3 ピン *2	LEFT1 LEFT2
	3		J3-4 ピン *1 J4-4 ピン *2	RIGHT1 RIGHT2
	4		J3-6 ピン *1 J4-6 ピン *2	TRGA1 TRGA2
	5		J3-7 ピン *1 J4-7 ピン *2	TRGB1 TRGB2
	6		キー配列指定入力 *4	H/L レベル(JIS・アイウエオ)
	7		CSAR (カセットテープリード)	
B	0	出力	J3-6 ピン *3	H レベル
	1		J3-7 ピン *3	H レベル
	2		J4-6 ピン *3	H レベル
	3		J4-7 ピン *3	H レベル
	4		J3-8 ピン *3	
	5		J4-8 ピン *3	
	6		ポート A 入力セレクト (ジョイスティック1、2の選択)	
	7		KLAMP (カナランプ信号「L」で点灯)	

注 意

*1 ポート Bのビット6が「L」レベル時に有効。ジョイスティック1用。
*2 ポート Bのビット6が「H」レベル時に有効。ジョイスティック2用。
*3 出力ポートとして使用しないときは「H」レベルにすること。
オープンコレクタバッファを介して出力する。
*4 JIS配列 ——「H」レベル、アイウエオ配列 ——「L」レベル

備 考

J3、J4の上記以外のピン端子信号

5ピン +5V

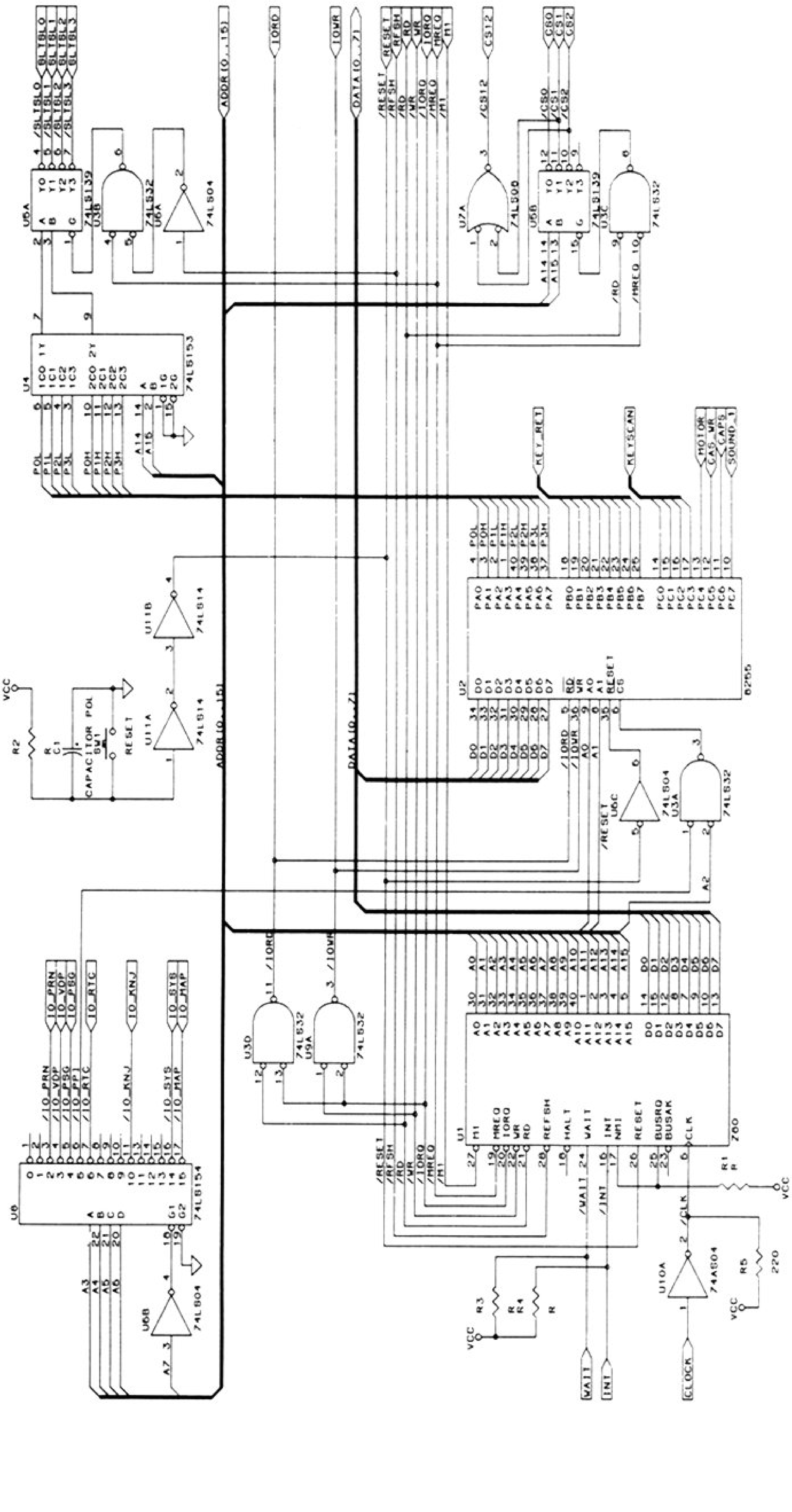
9ピン GND

7章

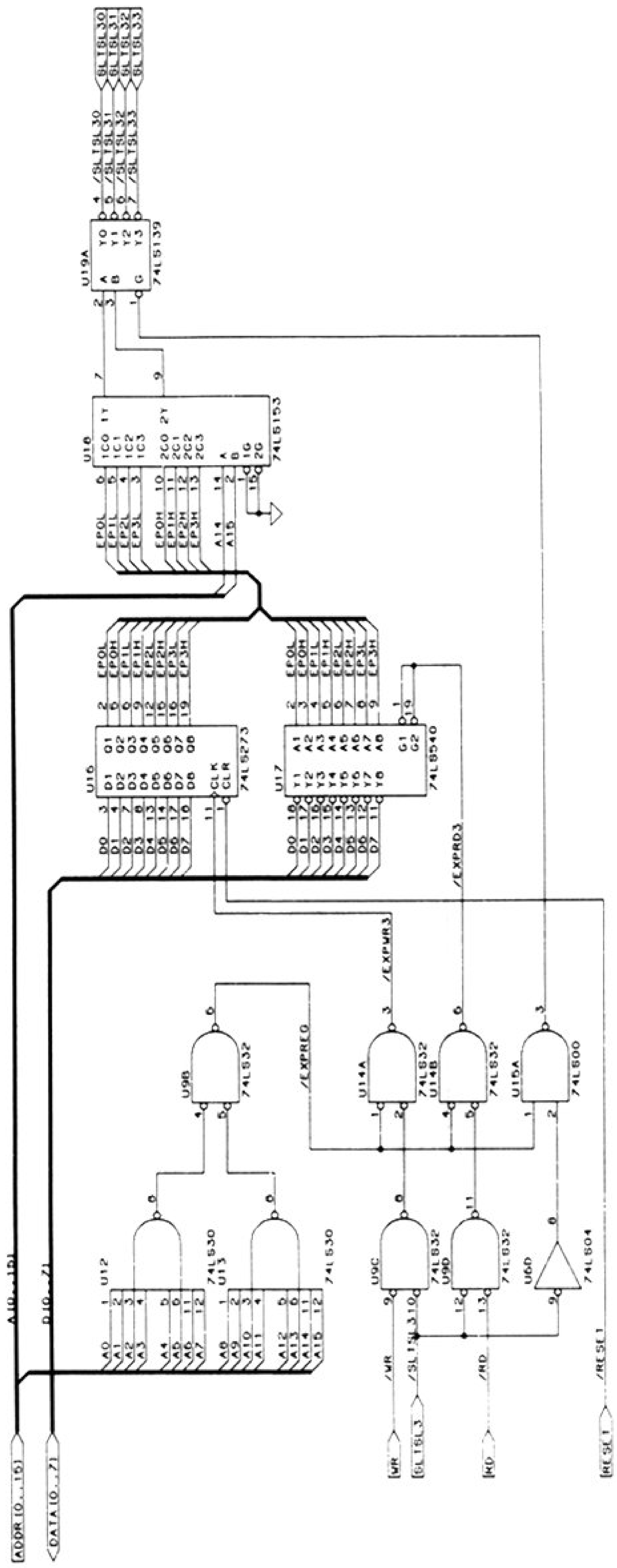
MSX2+回路例

この章では、MSX2＋の回路例を示します。この回路例は、MSX2＋の動作論理を理解するために参照して下さい。実際のMSX2＋では、周辺回路をまとめたLSIを使用していますが、この回路例では動作論理の理解という目的のために、あえてシステムロジックLSIを使用しない構成になっています。

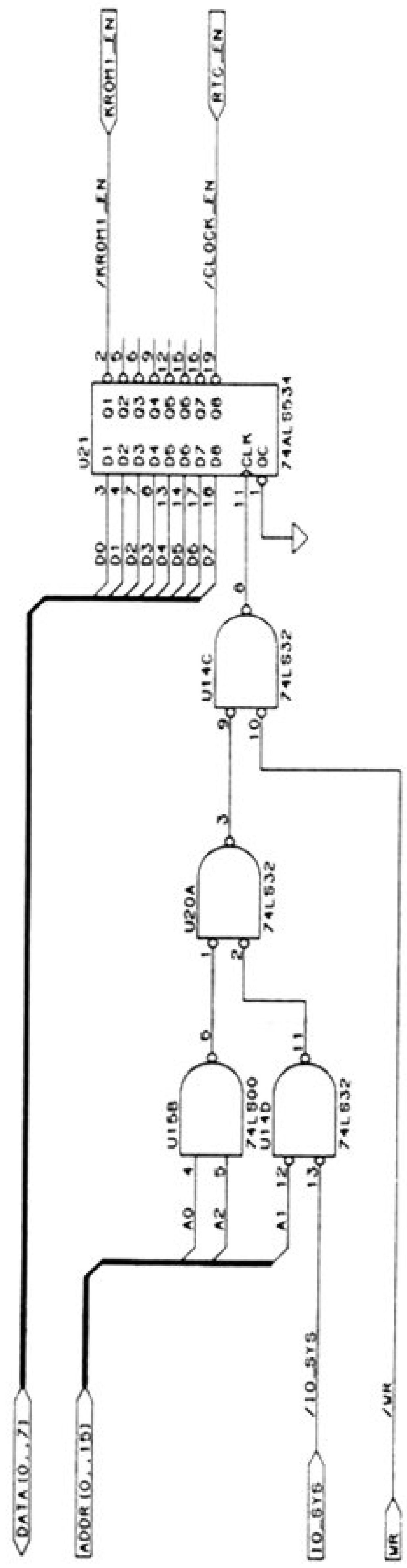
7.1 CPUと基本スロットセレクト信号



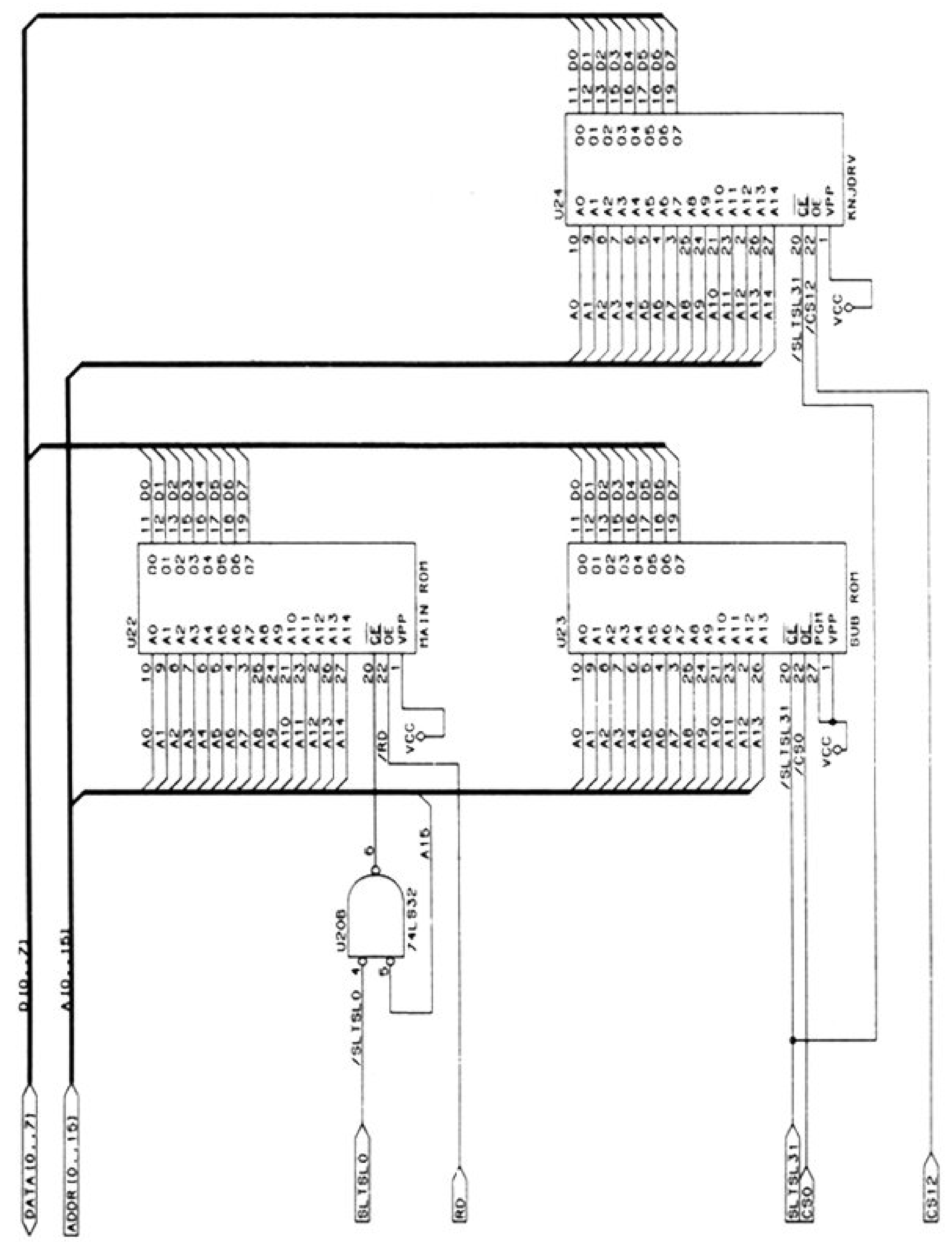
7.2 拡張スロットセレクト信号



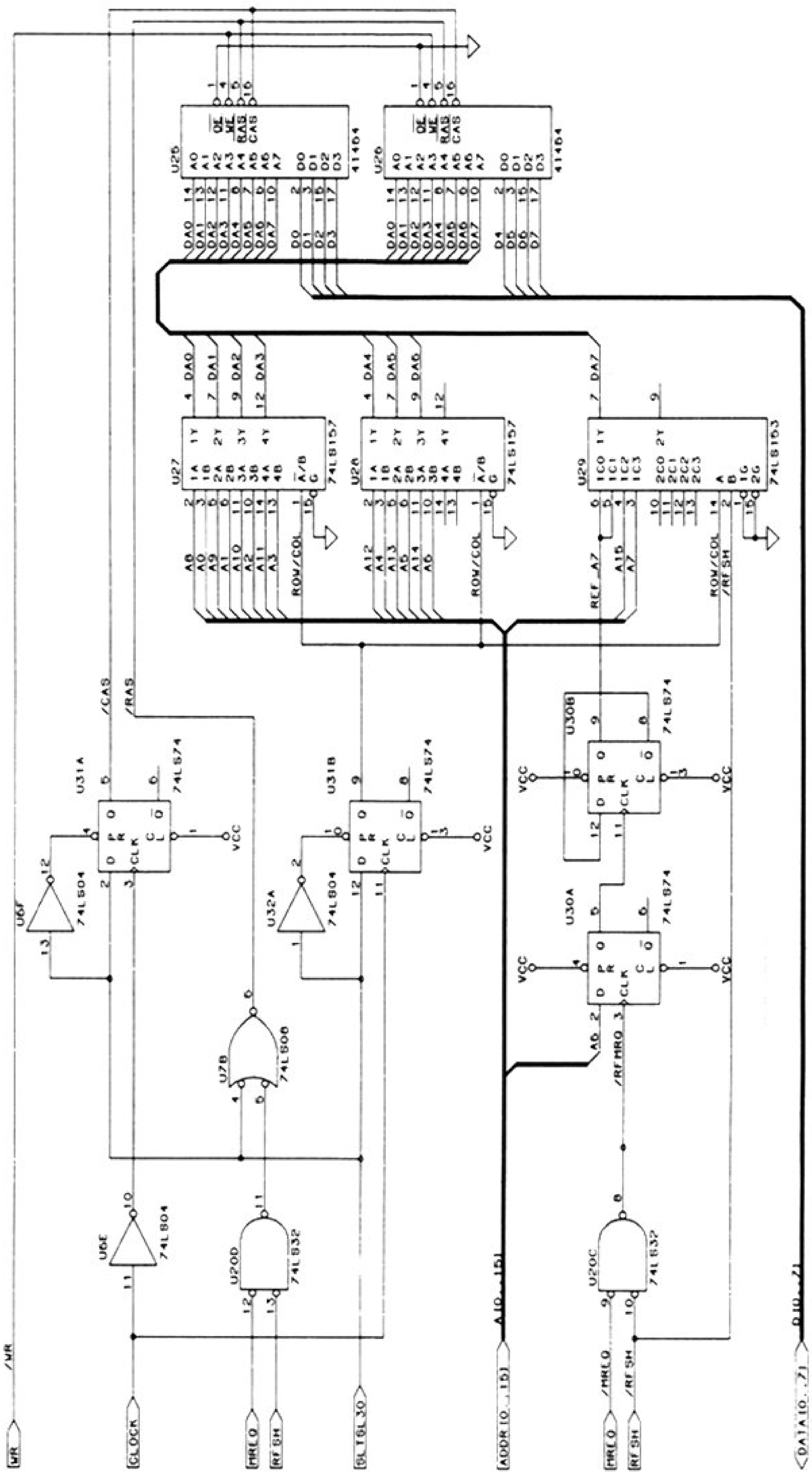
7.3 システムコントロールポート



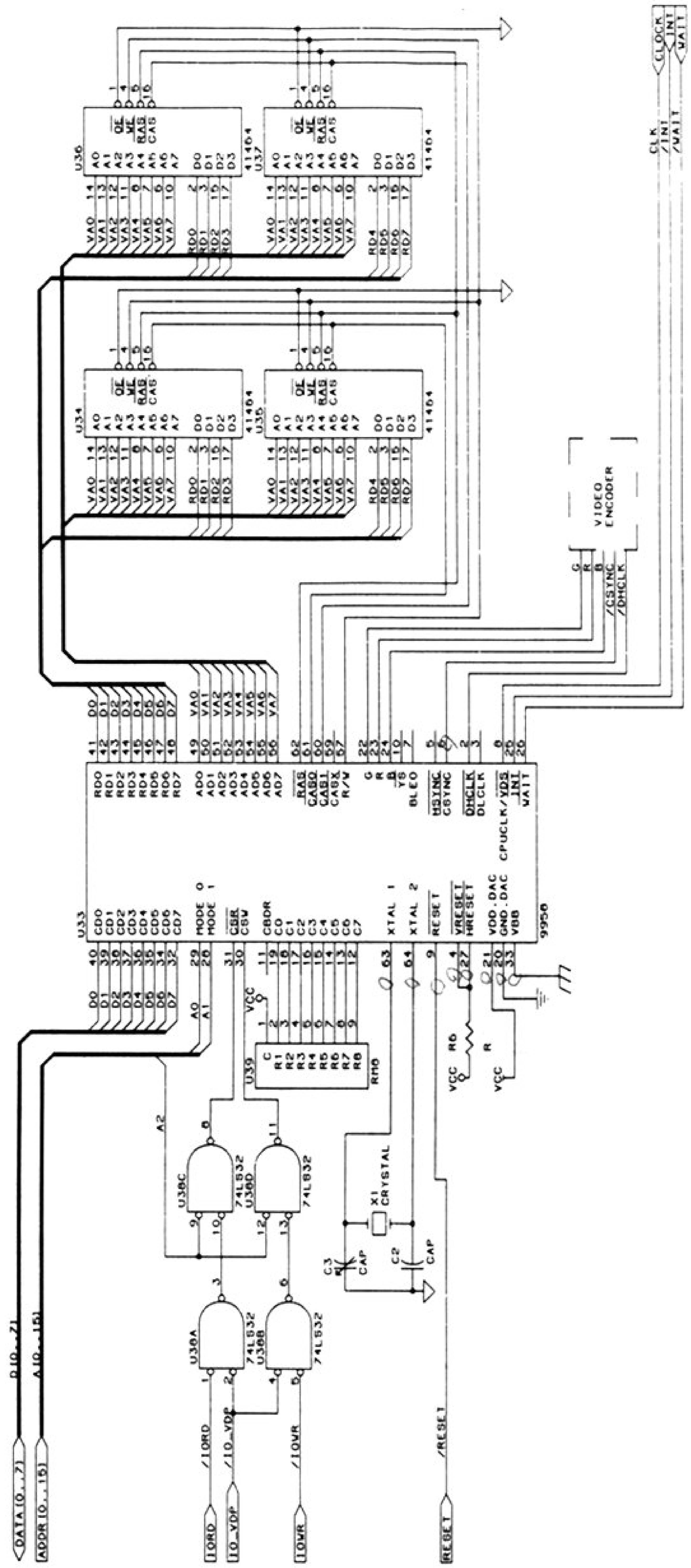
7.4 システム ROM


U24
KNJDRV
U22
MAIN ROM
U23
SUB ROM
U20B
74LS32
VCC
/SLTSL31
/CS12
/CS0
/RD
A15
/SLTSL0
DATA[0..7]
ADDR[0..15]
SLTSL0
RD
SLTSL31
CS0
CS12

7.5 メイン RAM

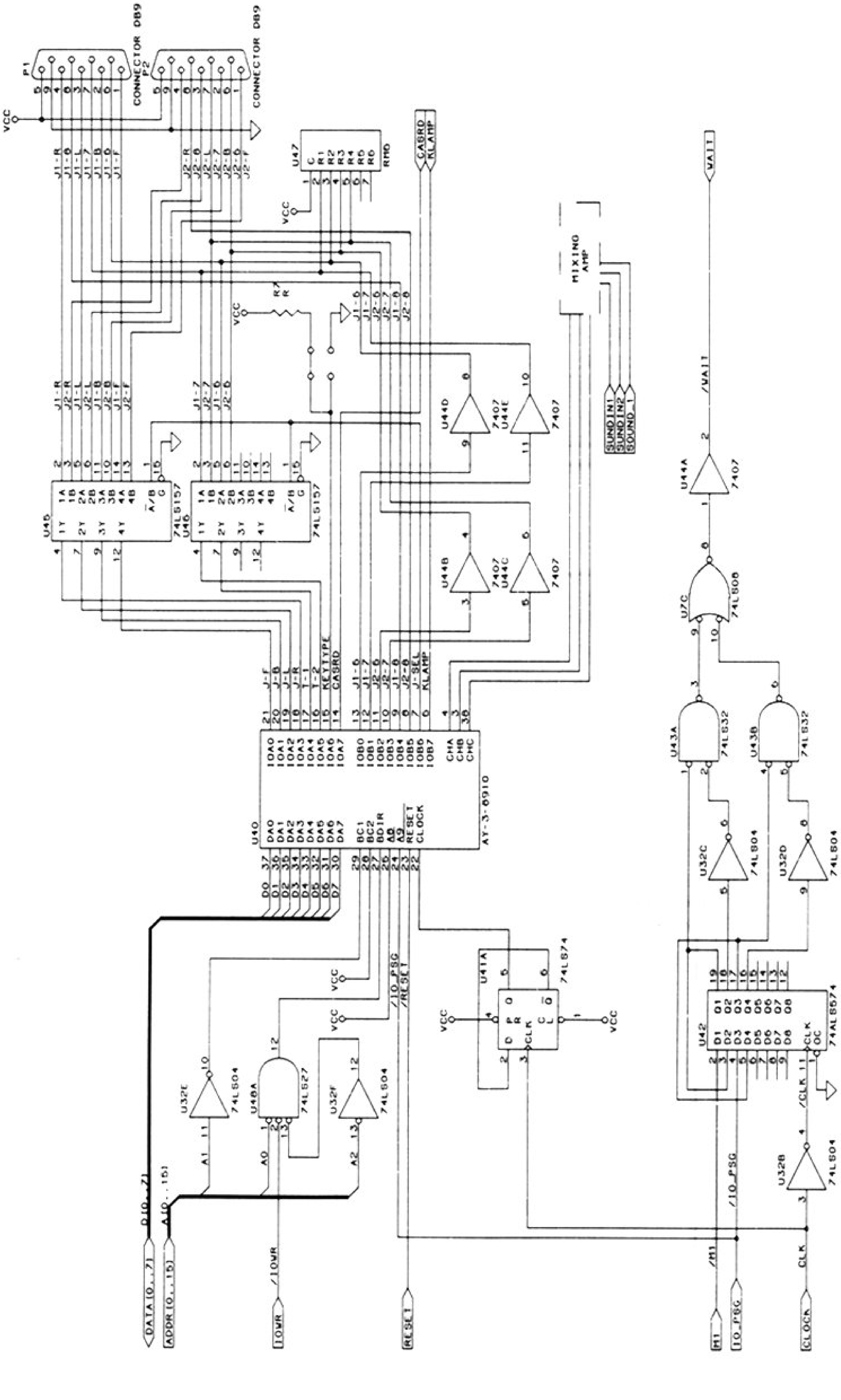

U25
41464
U26
41464
U27
74LS157
U28
74LS157
U29
74LS153
U31A
74LS74
U31B
74LS74
U30A
74LS74
U30B
74LS74
U6F
74LS04
U32A
74LS04
U6E
74LS04
U7B
74LS08
U20D
74LS32
U20C
74LS32
/CAS
/RAS
ROW/COL
/RFSH
REF A7
VCC
WR
CLOCK
MREQ
RFSH
SLTSL30
ADDR(0..15)
DATA(0..7)

7.6 VDP

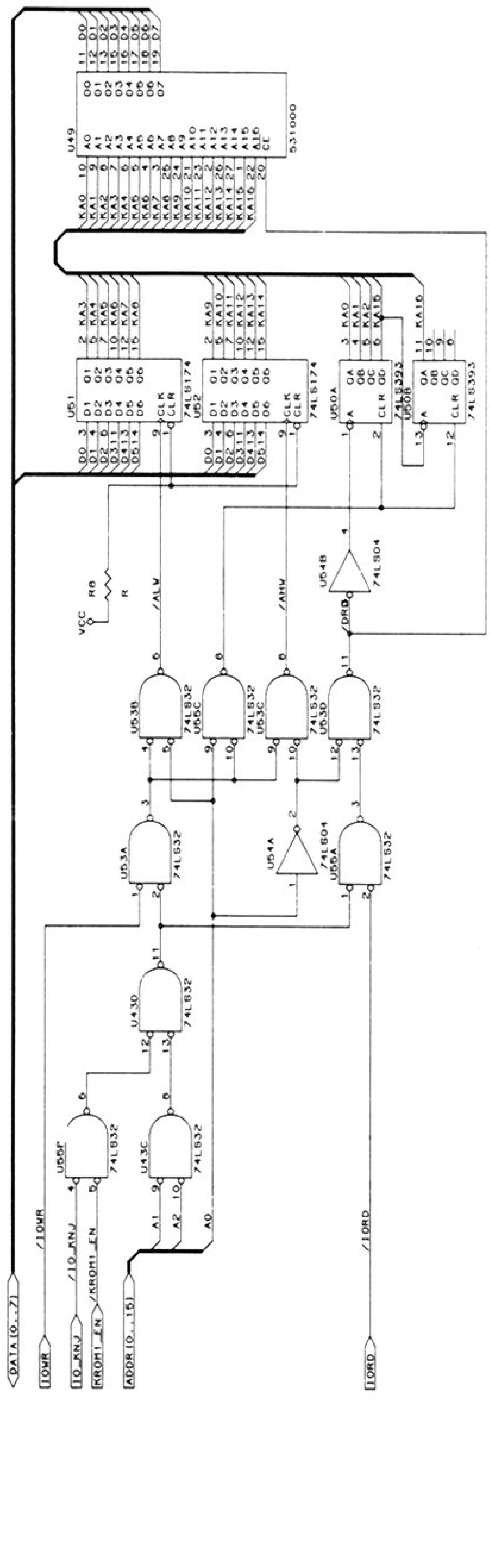


U33
9958
U34
U35
U36
U37
41464
VIDEO ENCODER
CLOCK
/INT
/WAIT
DATA(0..7)
ADDR(0..15)
IORD
IO_VDP
IOWR
RESET
X1 CRYSTAL
U38A
U38B
U38C
U38D
74LS32
U39
RM8

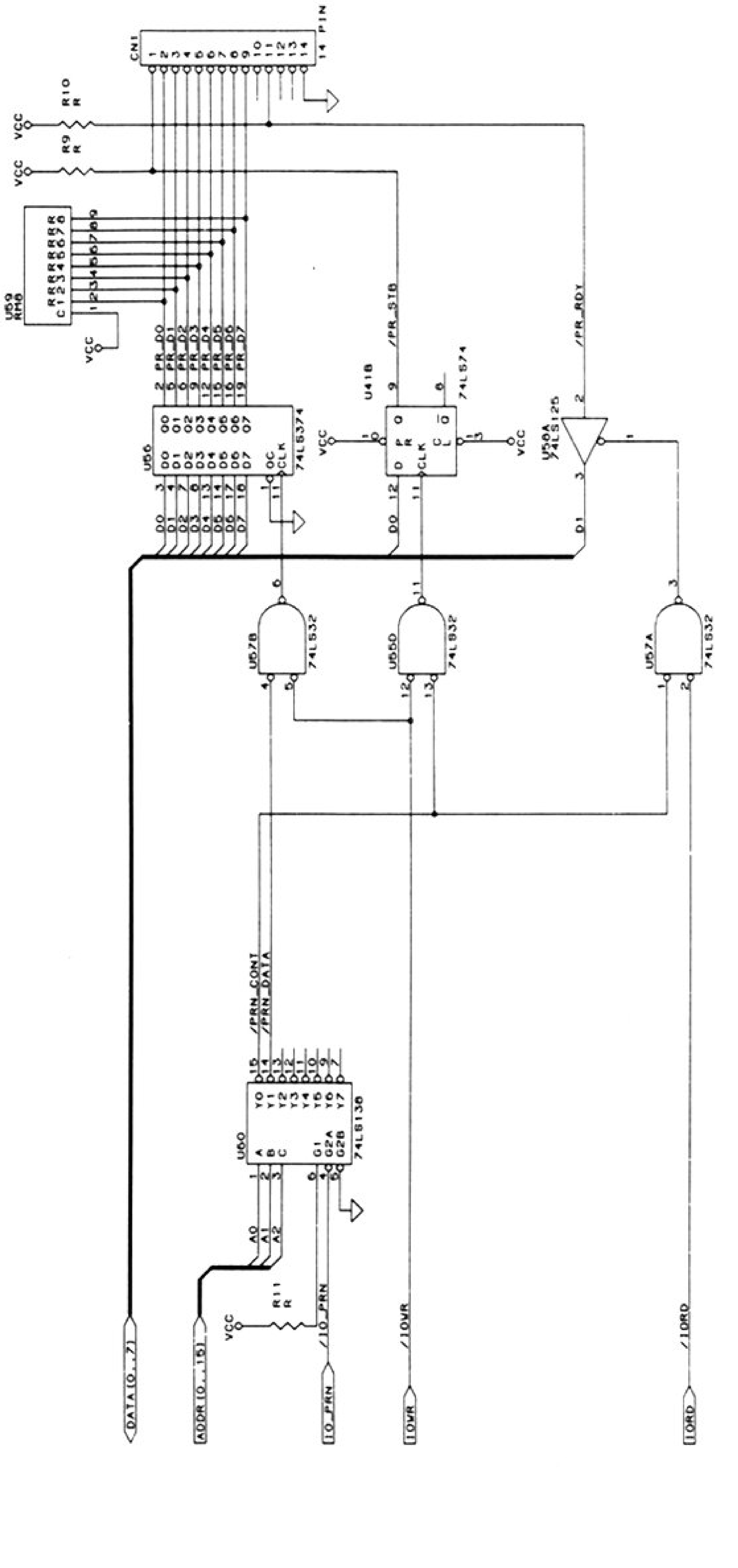
7.7 PSGとシステムウエイト



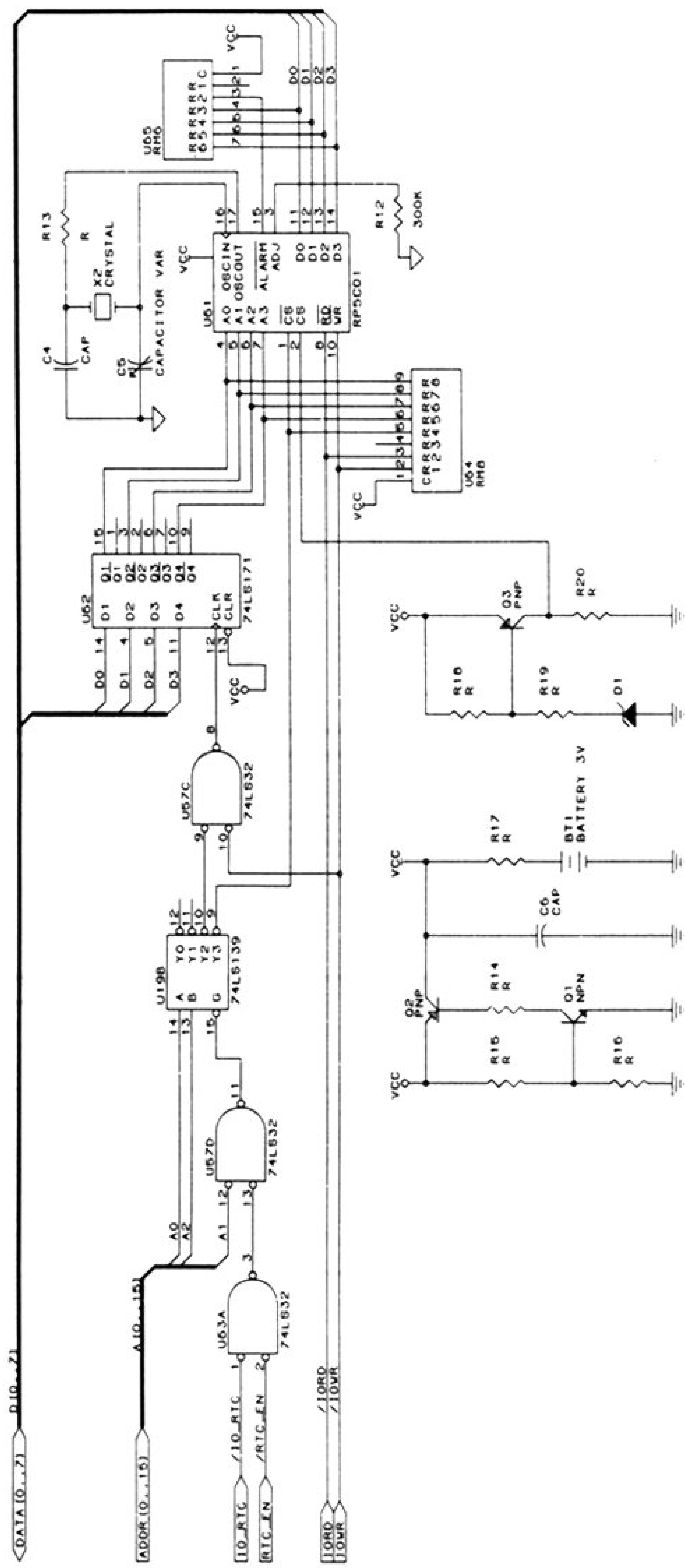
7.8 漢字 ROM



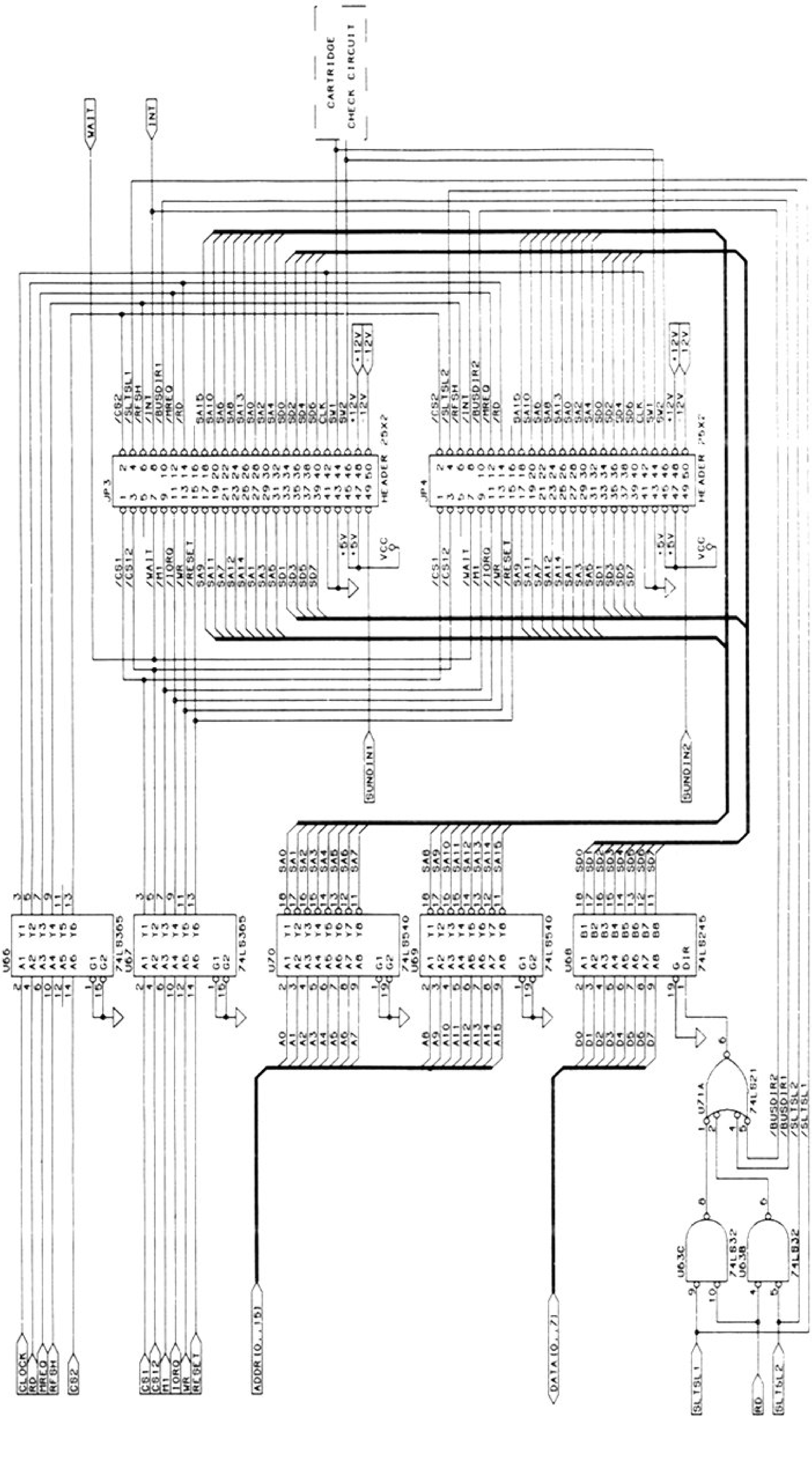
7.9 プリンタインターフェイス



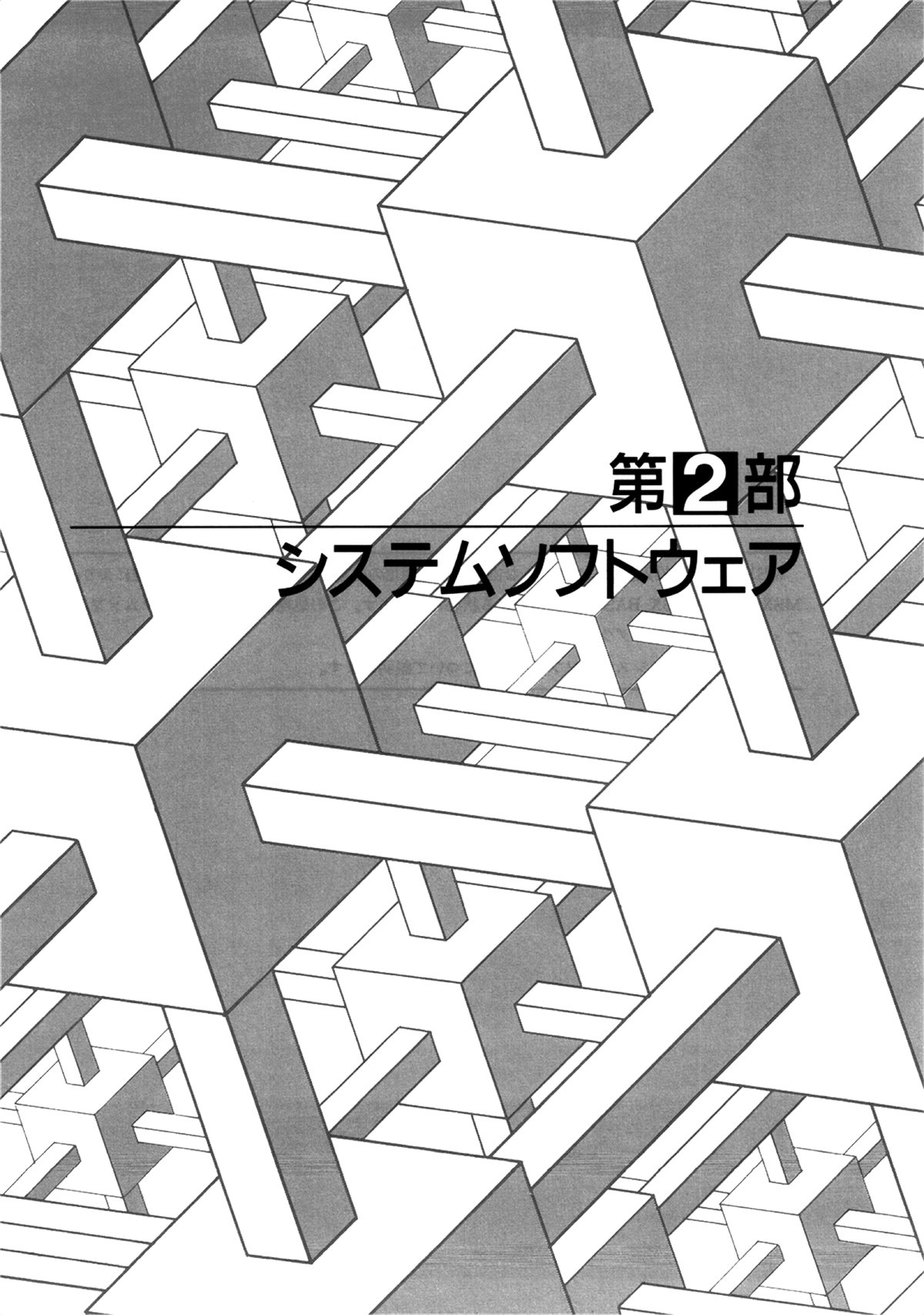
7.10 リアルタイムクロック



7.11 カートリッジスロット



第2部
システムソフトウェア



MSXでは電源投入後に、さまざまな内部的な初期化の処理を行っています。この処理により、MSX-DOSやMSX BASICが使用できる状態になります。この処理は、すべてシステムソフトウェアと呼ばれるプログラムが行います。

第2部では、このシステムソフトウェアについて説明します。

1章 ブートシーケンス

MSX2とMSX2+では、システムソフトウェアはMAIN ROMとSUB ROMに分かれています。また、ディスクを搭載したシステムの場合は、DISK ROMもシステムソフトウェアとして内蔵されています。

この章ではこれらのシステムソフトウェアが、実際に行う初期化の手順について解説します。

1.1 MAIN ROMでの初期化

MAIN ROMは下記の手順で初期化を行います。

1. マッパーおよびPPIの初期化
2. RAMのサーチ
3. システムワークエリアの初期化
4. SUB ROMのサーチ
5. ROMのサーチ
6. スクリーンの設定

1.1.1 マッパーおよびPPIの初期化

表 2.1　マッパーI/Oポートの初期値

I/Oポート	データ
0FFH	00H
0FEH	01H
0FDH	02H
0FCH	03H
0ABH	82H
0A8H	00H

ポート0FCHから0FFHはそれぞれページ0から3に現れるマッパーセグメントを指定するレジスタです。つまり、

ページ3	セグメント0
ページ2	セグメント1
ページ1	セグメント2
ページ0	セグメント3

が現れるように設定されます。

ポート0ABHはPPI(i8255)のモード設定レジスタで、ポート0A8Hが出力、ポート0A9Hが入力、ポート0AAHが出力に設定されます。

ポート0A8Hは基本スロットを選択するレジスタで、00Hを出力することにより、ページ0、1にMAIN ROMが現れることになります。

1.1.2 RAMのサーチ

MSXではRAMの存在するスロットを特に規定していないので、RAMのあるスロットを捜さなければなりません。RAMはページ2、3の順番で捜します。

ページ2のRAMは全てのスロットに対し、以下の順番でチェックします。

0BF00H から 0BFFFH まで1バイトずつ上位アドレス方向に
0BE00H から 0BEFFH まで1バイトずつ上位アドレス方向に
⋮
08000H から 080FFH まで1バイトずつ上位アドレス方向に

チェックは、

1. 1バイト読み出す
2. 1の補数を取る
3. その結果を書き込む
4. それを比較する
5. データを元に戻す

という手順で行いますので、元の内容が破壊されることはありません。途中でRAMが実装されていない（正しく比較されなかった）と判断された時点で、そのスロットに対するチェックは中断します。

これを全てのスロットに対して繰り返し、最大の容量を持つスロット（同じ容量のスロットが存在する場合は、スロット番号の若いもの）が選択され、ページ2に現れます。

基本スロットが拡張されたスロットであるかどうかの判断も、ページ2のRAMチェックと同時に行います。拡張スロットセレクトレジスタは、「書き込んだ値の1の補数が読み込まれる」と規定されていることを利用して拡張されているかどうかのチェックをしています。

ページ3のRAMも同じ様な手順で捜しますが、アドレスが以下のようになります。

0FE00H から 0FEFFH まで1バイトずつ上位アドレス方向に
0FD00H から 0FDFFH まで1バイトずつ上位アドレス方向に
⋮
0C000H から 0C0FFH まで1バイトずつ上位アドレス方向に

このようにして見つかったページ3のRAMがシステムのワークエリアとして使用されます。

1.1.3 システムワークエリアの初期化

ワークエリアの初期化はまず0F380Hから0FFF7Hまでを0でクリアすることから始まります。その後、「1.1.2. RAMのサーチ」で得られた拡張スロットに関する情報を【EXPTBL(0FCC1H〜0FCC4H)】と【SLTTBL (0FCC5H〜0FCC8H)】に書き込みます。その他の初期値はテーブルからブロック転送するかプログラムにより設定します。

次に、ページ2と3で連続して実装されているRAM容量のチェックを行い、【BOTTOM(0FC48H)】という変数を設定します。RAMチェックの順番は以下の通りです。このときも元の内容は破壊されません。

0FE00Hから0FEFFHまで1バイトずつ上位アドレス方向に
0FD00Hから0FDFFHまで1バイトずつ上位アドレス方向に
⋮
08000Hから080FFHまで1バイトずつ上位アドレス方向に

1.1.4 SUB ROMのサーチ

MSX2ではSUB ROMの存在するスロットは規定されていないので、SUB ROMを捜さなければなりません。SUB ROMは、

アドレスがページ0で、ROMのIDが「CD」の特殊なROMカートリッジ

です。普通のROMカートリッジと同様に、INITエントリ、STATEMENTエントリ、DEVICEエントリを持ちます。TEXTエントリは持つことができません。

SUB ROMでの初期化については、「1.2 SUB ROMでの初期化」で解説します。

1.1.5 ROMのサーチ

次に、MSXはROMカートリッジを捜しますが、MSX2+ (BASIC version 3.0) の場合はその前にMAIN ROMの1472H番地を経由します。ここには標準的には4バイトのNOP命令が書いてあるだけです。ハードウェアメーカーが他のスロットに優先して立ち上がるような内蔵ソフトを作りたいときには、このNOPを適当なインタースロットコール命令に置き換えれば実現

できることになります。

ROMはスロット番号の若い順に、4000H、8000H番地の順で、ROMのID(「AB」)が書いてあるかどうかを調べます。

ROM IDが見つかれば、INITエントリを調べ、0以外ならその番地をインタースロットコールします。もし、このROMがゲームのような、ROM内だけでループして動くソフトであったら、システムに制御が帰ってくることはありません。

次に、STATEMENTエントリ、DEVICEエントリ、TEXTエントリを調べ、0以外であったらそのスロットに相当する【SLTATR (0FCC9H〜0FD08H)】のそれぞれBIT5、6、7をセットします。

これを全てのスロットに対して繰り返します。

1.1.6 スクリーンの設定

クロックICにはバッテリでバックアップされたRAMが内蔵されていて、MSX2ではデフォルトのスクリーンモードなどが設定されています。ここではそのRAMの内容にしたがって各種のスクリーンモードを設定します。

1.2 SUB ROMでの初期化

SUB ROMの初期化プログラムはINITエントリから起動されます。その内容は以下の通りです。

1. システムワークエリアの初期化
2. システムコントロールポートの設定
3. クロックICの初期化
4. タイトルの表示

1.2.1 システムワークエリアの初期化

MAIN ROMの初期化では不十分な部分を初期化します。

1.2.2 システムコントロールポートの設定

MSX2 では、複数デバイスの競合による不都合を避けるため、標準的な内蔵デバイスに対して、その機能を禁止する機能があり、ポートの 0F5H がそれにあたります。内蔵デバイスはこのポートの対応するビットに 1 が書かれたときだけ機能するように設計されています。

表 2.2　システムコントロールポートの割り当て

ビット 0	第 1 水準漢字 ROM のためのビットです。
ビット 1	MSX2 では使用されていませんが、MSX2＋では第 2 水準漢字 ROM のためのビットに割り当てられています。
ビット 2	MSX-AUDIO のためのビットです。ポート 0C0H が 0FFH 以外であったら MSX-AUDIO が存在すると判定されます。
ビット 3	AV コントロールのためのビットです。ポート 0F7H と 77H を AND したものが 77H 以外だったら存在すると判定されます。
ビット 4	MSX　INTERFACE のためのビットです。ポート 0C8H が 0FFH 以外であったら MSX INTERFACE が存在すると判定されます。MSX INTERFACE は汎用の通信 LSI ですが、現在のところ使用されていません。
ビット 5	I/O マップされた RS-232C インターフェイスのためのビットです。i8251 をインターナルリセットした結果、ステータスレジスタが xx000101 (2 進数) になったら、存在すると判定されます。
ビット 6	ライトペンのためのビットです。ポート 0BBH が 0FFH 以外であったら存在すると判定されます。
ビット 7	クロック IC のためのビットですが、実際には使用されていません。

このように、漢字 ROM 以外のデバイスの判定は、MSX のバスがプルアップされている、つまり存在しないポートを読むと 0FFH が返って来ることを前提にしています。

1.2.3 クロック IC の初期化

クロック IC 内の RAM は初期化されていない場合もありえますので、ブロック 2(RAM-1)の最初の 1 ニブル (4 ビット) が 0AH でないか、WIDTH 情報が不正 (スクリーン 0 なのに 1 から 80 の範囲でない、またはスクリーン 1 なのに 1 から 32 の範囲でない)である場合は、RAM をデフォルトの内容で初期化します。デフォルトの内容は以下のとおりです。

表 2.3　クロック IC 内 RAM の初期値

ADJUST X	0
ADJUST Y	0
スクリーンモード	1
WIDTH の値	29
前景色	15
背景色	4
周辺色	7
カセットスピード	1200 ボー
プリンタモード	MSX
キークリック	あり
ファンクションキー表示	あり

1.2.4 タイトル表示

タイトル表示に関しては MSX2 と MSX2＋では大きく違いますので、それぞれの場合で解説します。

1. MSX2 の場合

タイトルを表示して(垂直スクロールを使用)、VRAM のサイズを調べます。サイズは VRAM の 1FFFFH 番地が読み書きできるかどうかで判定します。読み書きできたら 128KVRAM で、できなかったら 64KVRAM です。その後、VRAM サイズの表示、タイトル(MSX のロゴマーク)の表示、パスワードの表示と入力などを行います。

2. MSX2＋の場合

BIOS の RDRES を呼んでハードウェアリセットであるかどうかを調べます。
ハードウェアリセットでなければ、何もしません。

ハードウェアリセットなら、メイン RAM の総容量を調べ、RAM の ID を消します(これについては後で述べます)。MSX2＋では VRAM 容量は 128K に決まっているので調べません。その後、タイトルを表示して(水平スクロールを使用)、メイン RAM の総容量の表示、タイトルの表示、パスワードの表示と入力などを行います。

メイン RAM の総容量は以下のようにして求めます。

リスト 2.1　メイン RAM 容量の算出シーケンス

```
総容量=0
全てのスロットに対して{
        ページ1にそのスロットを出す
        セグメント0の第1バイトに0AAHを書く
        セグメント1の第1バイトに055Hを書く
        if セグメント0の第1バイト=0AAH{
                マッパー容量=0
                全てのセグメントに対して
                        第1バイトに0AAHを書く
                全てのセグメントに対して{
                        if 第1バイト<>0AAH
                                break
                        第1バイトに055Hを書く
                        if 第1バイト<>055H
                                break
                        マッパー容量=マッパー容量+1
                }
                総容量=総容量+マッパー容量
        }
}
```

上記の方法で、マッパー上に存在する全てのRAMの総セグメント数が計算できます。表示されるメインRAM総容量は、総セグメント数の16倍ですが、0の場合は特別に64Kとみなして表示します。これはマッパー回路に接続されていない64KのRAMが存在する場合を考慮したためです。

RAMのIDの消去は全てのスロットの4000H、8000H番地と、選択されているRAMの0C010H番地に0を書くことにより行われます。これはページ1のRAMにROMのイメージを転送して動作するようなプログラムが、電源をON・OFFしても動いてしまうことを防止するためです。DRAMの中には、電源を切っただけでは内容がなかなか消えないものがあるためこのような方法を取っています。また、0C010Hに0を書くのは、ここをフラグとして使用しているROMのプログラムがあったためです。

1.3 FDD ROMでの初期化

ここでは、ディスク環境での立ち上がりシーケンスについて解説します。
ただし、ここで解説するのはDOS1の初期化についてだけです。DOS2は限られた紙面で解説するにはあまりにも複雑で、マスターカートリッジ（後述）を乗っ取ってマスターになるなど本質的なこと以外の部分が多く、理解の妨げになるためです。

また、この章の内容は、DOSのバージョンアップに伴って変更される可能性がありますので(実

際に DOS2 では違う)、あくまでも内部動作の理解のために使用して下さい。商用のアプリケーションソフトウェアでは、この内部情報は使用しないで下さい。

ディスクカートリッジは、MSX にとって特別な地位を約束されているわけではなく、一般のカートリッジと同等の扱いを受けます。ただし、システムに深く密着したソフトウェアですので、MSX で規定されていない RAM のワークエリアなどを使用しています。

以下にディスクカートリッジの INIT エントリで実行される初期化シーケンスを記述します。ここで INIHRD、DRIVES、INIENV はディスクカートリッジ内のディスクドライバのルーチンです。alloc は指定されたバイト数だけ上位アドレスから割り当てそこへのポインタを返す内部ルーチン、DISKID は 0FD99H 番地の内容で初期値は 0、p はポインタ変数、f はフラグ変数です。

DEFDPB はドライバ内部にある定数テーブルで、少なくともそのテーブルによって、ドライバがサポートしているすべてのメディアのブートセクタおよび FAT の第1セクタが読めることが保証されています。【$DPBLIST (0F355H)】は DPB へのポインタを要素とする配列です。

リスト 2.2 DISK ROM の初期化シーケンス

```
INIHRD()                /*  ディスクハードウェアの初期化  */
if DISKID>=80H
        return
if DISKID=0 {
        if SHIFT キーが押されている {
                ビープ
                DISKID=0FFH
                return
        }
        p=alloc (固定ワークエリアサイズ+ドライバワークエリアサイズ)
        固定ワークエリアを初期化
        f=CONTROL キーが押されている
        ビープ
        H.RUNC の設定
} else
        p=alloc (ドライバワークエリアサイズ)
p を SLTWRK にリンク
n=DRIVES (f)            /*  ドライブ数を返す  */
DRVTBL に設定
p=alloc (n*21)
そのカートリッジ内の全てのドライブに対して {
        DEFDPB を p に転送
        p を$DPBLIST にリンク
        p=p+21
}
INIENV()                /*  ディスクドライバの環境初期化  */
DISKID=DISKID+1
```

DISKIDはディスク環境のディスエーブル、および最初のディスクカートリッジであるかどうかの判定に使われます。最初のディスクカートリッジはマスターカートリッジと呼ばれ、ディスク機能の大部分を受け持ちます。他のディスクカートリッジはスレーブカートリッジと呼ばれ、ディスクドライバの部分だけ使用されます。

マスターカートリッジは【H.RUNC (0FECBH)】というフックを設定します。MSXが全てのカートリッジのスキャンを終えて、BASICを起動する際に必ず呼ばれるフックです。H.RUNCは以下に示すようなルーチンに接続されます。

【H.STKE (0FEDAH)】はディスクの機能を使用するROMカートリッジが設定するフックです。

リスト2.3　H.RUNCが接続されているルーチン

```
変数の初期化
ドライバ用セクタバッファ          = alloc（最大セクタサイズ）
DOS汎用セクタバッファ             = alloc（最大セクタサイズ）
DOS ディレクトリ用セクタバッファ  = alloc（最大セクタサイズ）
全てのドライブについて {
        p = alloc（セクタサイズ＊FAT サイズ＋1）
        p＋1 をそのドライブの DPB 内の FAT ポインタにリンク
}
スクリーンの設定
ページ 0、1 の RAM を捜す
SP = 0C200H
if H.STKE が設定されている
        BASIC ROM へジャンプ
if TEXT エントリを持ったカートリッジがある
        BASIC ROM へジャンプ
ブートセクタを読み最初の 256 バイトを 0C000H に転送する
if ブート不可能
        DISK BASIC を起動する
0C01EH をキャリーフラグをリセットして呼ぶ
if RAM が 64K ない
        DISK BASIC を起動する
ページ 0 の RAM を表に出す
ページ 0 を初期化する
スロットハンドラを設定する
割り込みハンドラを設定する
ブートセクタを読み最初の 256 バイトを 0C000H に転送する
if ブート不可能
        DISK BASIC を起動する
0C01EH をキャリーフラグをセットして呼ぶ
```

リスト2.3の6行目でallocされるメモリが、FATに必要とされるより1バイト多く割り当てられていることに注意して下さい。FATの先頭の1バイト前はメモリ内のFATとディスク上のFATが一致しているかどうかのフラグです。ディスク上のFATがまだメモリ内に読み出されていないときには255が、メモリ内のFATのみを変更して、まだディスクに書き出していないときには1が、一致しているときには0が書かれています。

ディスク環境では全てのページのRAMのスロットアドレスがページ0から順に0F341H、0F342H、0F343H、0F344Hに格納されます。ページ2、3のRAMはMAIN ROMによって既に捜されていますから、それをスロットアドレスに変換して格納します。ページ0、1のRAMはマスターカートリッジが捜します。

ページ0のRAMは全てのスロットに対して、

0010Hから0001Hまで1バイトずつ下位アドレス方向に
0410Hから0401Hまで1バイトずつ下位アドレス方向に
⋮
3C10Hから3C01Hまで1バイトずつ下位アドレス方向に

ページ1のRAMは全てのスロットに対して、

4010Hから4001Hまで1バイトずつ下位アドレス方向に
4410Hから4401Hまで1バイトずつ下位アドレス方向に
⋮
7C10Hから7C01Hまで1バイトずつ下位アドレス方向に

の順でチェックされます。これも、MAIN ROMでのチェックと同様、元の内容を破壊することはありません。1Kバイトおきに15バイトだけがチェックされることに注意して下さい。RAMが実装されていないと判断された時点でそのスロットに関するチェックは終了し、16Kフルに実装されているスロットが見つかった時点で全てのチェックが終了します。

INITエントリおよびマスターカートリッジによる初期化が終了した時点で、RAMの内容は以下のようになっています。ここではディスクカートリッジが2つあって、最初のカートリッジは2台、次のカートリッジは1台のディスクをサポートしているとします。



図 2.1　マスターカートリッジ初期化終了時のワークエリア

【HIMSAV（0F349H)】はディスクシステムの核（カーネル）が動作するのに最低限必要なエリアの底へのポインタです。【$SECBUF（0F34DH)】、【$BUFFER（0F34FH)】、【$DIRBUF（0F351H)】は 3 種類のセクタバッファに対するポインタです。

2章
割り込み

この章では、MSX の割り込みについて解説します。

MSX は Z80 の「モード 1 割り込み」と呼ばれる割り込みを使用しています。これは、割り込みがかかると必ず 38H 番地をコールするという割り込みモードです。MSX の MAIN ROM 内の割り込みルーチンで処理していることを、以下に解説します。

リスト 2.4　割り込みシーケンス

```
全てのレジスタ（裏レジスタ、IX、IY 共）をセーブ
H.KEYI を呼ぶ
ライトペンの処理
VDP のステータスレジスタを読む
if 垂直帰線割り込み {
        H.TIMI を呼ぶ
        割り込みの許可
        STATFL の設定          /＊ 【STATFL (0F3E7H)】 ＊/
        ON SPRITE の処理
        ON INTERVAL の処理
        JIFFY = JIFFY＋1       /＊ 【JIFFY (0FC9EH)】 ＊/
      PLAY 文の処理
        SCNCNT = SCNCNT－1    /＊【SCNCNT (0F3F6H)】 ＊/
        if SCNCNT = 0 {
                SCNCNT = 2
                ON STRIG の処理
                キーボードのスキャン
        }
}
全てのレジスタ（裏レジスタ、IX、IY 共）をリストア
```

MSX の割り込み源のうち、システムが管理するのは垂直帰線割り込みだけです。周期は NTSC 版の場合 1/60 秒、PAL/SECAM 版の場合 1/50 秒です。日本版は NTSC ですから、1/60 秒になります。

また、VDP のステータスレジスタを読む際には、単にポート 099H を読むだけですから、ステータスレジスタポインタが 0 以外だと、割り込みがかかりっぱなしになり、MSX は暴走することに

なります。ステータスレジスタポインタを0以外に変更するときには必ず割り込みを禁止してください。そして、0に戻してから割り込みを許可してください。

【JIFFY (0FC9EH)】は垂直帰線割り込み毎にインクリメントされる1ワードの変数ですので、時間をカウントするのに使用することができます。

【SCNCNT (0F3F6H)】はキーボードおよびジョイスティックのスキャンを間引くためのものです。MSXでは2回の割り込みで1回スキャンされることになります。

ON STRIGの処理を行うためジョイスティックがスキャンされますが、このときジョイスティックポート1、2ともに、6、7番ピンがHIGHに、8番ピンがLOWになります。ジョイスティックポートに接続するための周辺機器を設計して、これらのピンを出力ピン（MSXから見て）として使用する際、このことに充分注意してください。

具体的には6、7番ピンはHIGHが、8番ピンはLOWが定常状態となるように設計して、アクセスするときは割り込みを禁止して行う設計にしてください。

【H.KEYI (0FD9AH)】は割り込みがかかると必ず呼ばれるフックです。垂直帰線割り込み以外の割り込みを処理するようなルーチン（RS-232Cや水平帰線割り込みなどのハンドラ）は、このフックを使用します。【H.TIMI (0FD9FH)】は垂直帰線割り込みがかかると呼ばれるフックです。

割り込みフックの使用法には次の2種類があります。

1. JP命令を書く

ゲームなどのプログラムでは、一番効率の良い方法です。

前提条件としては、いつ割り込みがかかっても割り込みハンドラがCPU空間に出ていなければならないことです。つまり割り込みハンドラが「スロットが切り換わる可能性のあるページ」にいてはならないことになります。

割り込みハンドラからリターンするときには、

1) そのまま「RET」命令を実行する
2) スタックを1段ポップして、すべてのレジスタをリストアして「RET」命令を実行する
3) セーブしてある、以前のフックにジャンプする

などの方法があります。

1)では、MAIN ROMの割り込み処理が続行しますので、上で述べた処理（キーボードのスキャンなど）が必要なときは、この方法を採用します。ただし、割り込みフックを設定する前に、他のプログラムが既にこのフックを使用していたときは、それは無視されます。

2)では、MAIN ROMの割り込み処理は完全に無視されます。最高の効率が必要な場合はこの方法を採用します。ただし、MAIN ROMの割り込み処理が前提となっているような機能（BIOSのCHSNS、CHGETなど）のサービスは受けられません。また、以前の割り込みフックは無視されます。

3)では、MAIN ROMの割り込み処理、以前の割り込みフックともに、実行されます。具体的には、フックを設定する前に、5バイトのセーブエリアにフックの内容をコピーして、それからフックを設定することが必要です。割り込みハンドラが終了したら、このセーブエリアにジャンプします。これを割り込みのチェーンと呼びます。

2. RST 30H（FAR CALL）命令を書く

システムの一部として組み込まれる機能（例えば、RS-232C）を処理する割り込みハンドラは必ずこのようにしなければなりません。システムとして動いているときは、スロットが頻繁に切り換わるためです。

また、割り込みのチェーンも完全に行わなければなりません。よく、「ディスクのアクセスランプが光ったままになってしまう」という現象がありますが、これは割り込みのチェーンを行っていないプログラムが動いているからです。ほとんどのディスクドライバは垂直帰線割り込みを使用して、モーターのON・OFF（アクセスランプのON・OFF）を行っています。

添付のフロッピーディスクに、「INT.MAC」という名前のサンプルプログラムが入っています。

3章 BASIC

3.1 MSX BASIC

MSX のシステムに標準で搭載される言語は BASIC です。何度か拡張、改良され現在次のようなバージョンがあります。

MSX BASIC ver. 1.0	MSX1 の BASIC
MSX BASIC ver. 2.0	MSX2 の BASIC
MSX BASIC ver. 3.0	MSX2＋の BASIC
MSX Disk BASIC ver. 1.0	MSX-DOS1 下の Disk BASIC

MSX BASIC ver 2.0 では MSX2 で改良されたグラフィックス、バッテリーバックアップされたメモリの利用などのために機能が追加、拡張されています。

MSX BASIC ver 3.0 では MSX2＋で拡張されたグラフィックや漢字処理機能をサポートするための機能が追加、拡張されています。

3.2 BASIC の動作モード

MSX-DOS がない MSX のシステムでは、直接アプリケーションを実行する場合を除き BASIC、あるいは Disk BASIC が起動します。MSX-DOS のプロンプトから「BASIC」と入力して起動することもできます。BASIC は、ダイレクトモードとプログラムモードの 2 つの動作モードで実行されます。

3.2.1 ダイレクトモード

BASICの命令文だけをキー入力すると、その文はリターンキーを押した後すぐに実行されます。これをダイレクトモードでの実行といいます。ダイレクトモードで入力された命令はメモリには記憶されません。

3.2.2 プログラムモード

BASICの命令文の前に番号をつけてキー入力すると、その行はプログラム内の行として、番号とともにコンピュータのメモリに記憶されます。この番号のことを行番号といい、行番号の付いた命令文の集まりをプログラムといいます。プログラムはRUN命令やGOTO命令などを使って実行させることができます。これをプログラムモードでの実行といいます。

1. プログラムの作成

プログラムとは、行の集まりです。行番号は0〜65529の整数で、できたプログラム行は行番号の小さい順に実行されます。BASICの命令はアルファベットの大文字で入力しても小文字で入力しても、自動的に大文字に変換されて処理されます。ただし、引用符（"）で囲まれた文字や、REMやアポストロフィー（'）に続くコメント文は、そのまま記憶されます。

2. プログラムの修正

プログラムの修正は、以下のようにします。

■行の追加

追加したい内容に、追加したい行番号をつけて入力します。入力と同時に行は番号順に並べ換えられて記憶されます。

■行の削除

削除したい行の行番号だけを入力すると、その行は削除されます。まとめて削除するときには、DELETE命令を使います。

■行の置き換え

訂正したい行の行番号と新しい内容をキー入力してリターンキーを押せば、古い行の内容は自動的に新しいものに変わります。また、LIST命令を使ってプログラムを画面に表示し、カーソルを直したい所に移動して直接書き換えてからリターンキーを押しても、その行は訂正されます。

MERGE命令によって、ファイルから新しい行を読み込むこともできます。

■行番号のつけかえ

行番号をつけかえたいときには、RENUM命令を使います。

3.3 BASICの文法

3.3.1 BASICで使用できる文字と特殊記号

MSX BASICで使用できる文字は、英文字（大文字・小文字)、数字、カナ文字、ひらがな、特殊記号、そしてグラフィック文字です。文字についての詳細は、巻末の「キャラクタコード表」を参照してください。また、漢字BASICのモードではシフトJISコードによって全角漢字を処理することができますが、この場合は半角ひらがなは扱いません。

1. 特殊記号

使用できる文字の中でも、特殊記号と呼ばれる文字はそれぞれ特別な意味を持っています。特殊記号のうち、演算子については、「式と演算」で説明します。

.（ピリオド）　現在BASICが着目している行番号を表します。

例）プログラム実行中にエラーが発生したとき、

LIST.

と入力すると、エラーの発生した行が表示される。

-（マイナス）　数値の範囲を指定するときに使います。

例）LIST命令で表示する行の範囲を指定するとき、

LIST 100-200

とすれば、100行から200行までが表示される。

:（コロン）　マルチステートメントの区切りとして使います。

例）A=B+C:PRINT A

,（カンマ）　データを複数個指定するときに、その区切りとして使います。PRINT命令で使われた場合には、偶数番目のデータは左から15桁目に表示されます。

例）INPUT A,B,C

；(セミコロン)　データを複数個指定するときに、その区切りとして使います。PRINT命令で使われた場合には、データは前のデータに続いて表示されます。

例) PRINT A；B；C

'(アポストロフィ)　REM命令の代わりに用いて、注釈文を作ります。

例) 'コノキ゛ョウハ　コメントテ゛ス

？(疑問符)　PRINT命令の代わりに用います。

例) ？ 25＊12

(空白)　プログラムを見やすくするために、文中に好きなだけ空白を入れることができます。ただし、予約語（巻末の「予約語表」参照）の中には空白を入れることはできません。また、文字定数（「定数」参照）の中の空白は文字としての意味を持っています。文中の空白はプログラムの実行には影響を与えませんが、メモリには空白も記憶されます。

3.3.2 BASICで扱うデータ

BASICで扱うデータは、次のように分類することができます。

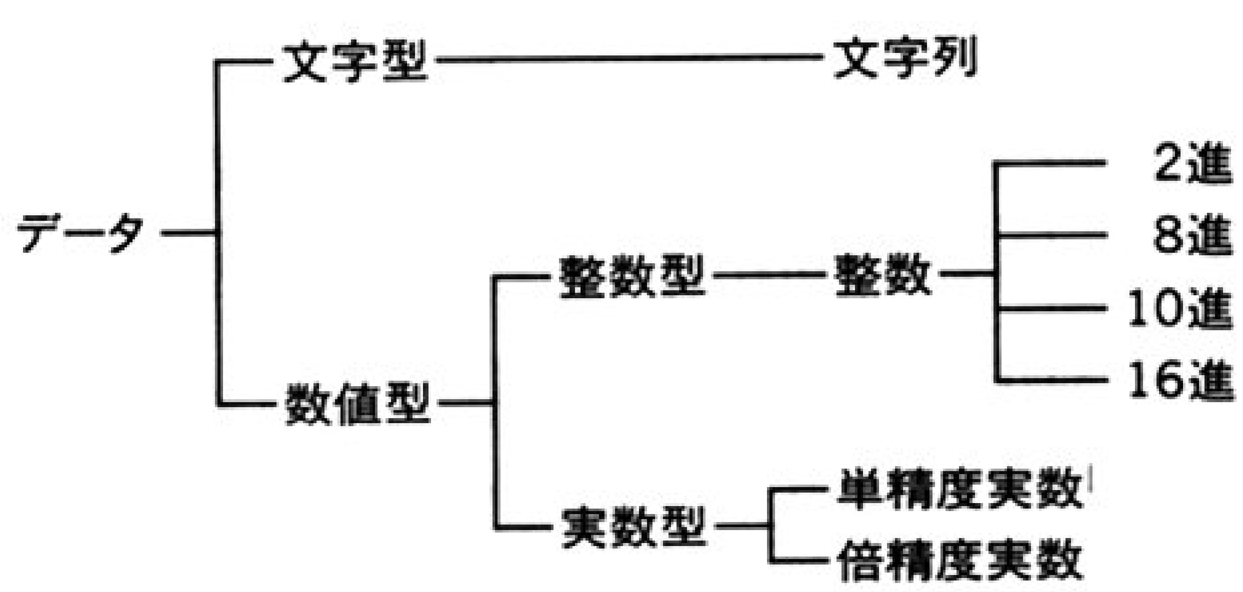


図2.2　BASICで扱うデータ

1. 文字列

255文字以内の文字をつないだデータです。文字列には、BASICで使用できる文字ならばどれでも使うことができますが、文字列を算術演算に使うことはできません。

2. 数値

数値は算術演算を行うことができるデータです。数値には整数型と実数型とがあり、数値の取る値はそれぞれで異なります。

3. 整数

−32768から＋32767までの整数です。整数型の数値を使う演算は実行時間が短くてすみます。

4. 実数

実数には有効桁数によって、単精度実数と倍精度実数とがあります。MSX BASICでは、実数を使った演算はすべて倍精度実数に変換して行われ、結果も倍精度実数で得られますが、単精度実数として記憶したり、結果を単精度実数で受け取ることもできます。

5. 単精度実数

有効桁6桁の実数です。単精度実数で表せる数値は−9.99999E＋62から9.99999E＋62で、指数の範囲はE＋62〜E−64までです。

6. 倍精度実数

有効桁14桁の実数です。倍精度実数で表せる数値は−9.9999999999999E＋62から＋9.9999999999999E＋62で、指数の範囲はE＋62〜E−64までです。

また、数値型データは必要に応じて、その型を他の型に変換することができます。型の変換は、次の規則にしたがって行われます。

- ■数値を異なる型の数値変数に代入する場合、数値は代入先の変数の型に変換される。
- ■精度の異なる数値を使った演算では、精度の低い方が高い方の型に変換されて演算が行われる。
- ■論理演算の場合、扱われる数値はすべて整数型に変換され、結果は整数で得られる。
- ■実数が整数に変換される場合は、小数点以下は切り捨てらる。このとき、切り捨てた結果が整数型で扱える範囲（−32768〜32767）を越えていればエラーとなる。
- ■倍精度実数型変数を単精度実数型変数に代入すると、変数の値は有効数字6桁に直されたものになる。もとの倍精度実数型の7桁目以降は、四捨五入される。
- ■文字型と数値型のデータの変換は、VAL、STR$関数を使う。

3.4 定数

定数とは、それぞれに固有の値を持ったデータのことです。定数の書き方は、先に述べたデータの型によって異なります。

3.4.1 文字定数

文字定数は、文字列を引用符（"）で囲んで表します。文字列の長さは255文字以内でなければなりません。また、数字を引用符で囲むと文字列とみなされ、算術演算に用いることはできません。

例

"THANK YOU"
"3.14159265"

3.4.2 整数型定数

整数型の定数には、以下の 4 つの形式があります。

1. 10進形式

−32768〜+32767 までの整数です。負の整数の場合、数値の前に必ずマイナス符号（−）をつけなければなりませんが、プラス符号は省略できます。また、−32768〜+32767 までの実数の後ろに%をつけると、小数点以下は切り捨てられて整数型の定数とみなされます。

例

0
−123
3875.6%

2. 8進形式

先頭に&O をつけて 8 進数 (0〜7) を並べたものです。範囲は&O0〜&O177777 までですが、&O177777〜&O100000 までは負の数で、−1 から −32768 に対応します。

例

&O012345　　10進数では5349
&O177777　　10進数では−1

3. 16進形式

先頭に&Hをつけて16進数（0〜9、A〜F）を並べたものです。範囲は&HO〜&HFFFFまでですが、&HFFFF〜&H8000までは負の数で、−1から−32768に対応します。

例

&H100　　10進数では256
&HFFFF　　10進数では−1

4. 2進形式

先頭に&Bをつけて0と1を並べたものです。範囲は&B0〜&B1111111111111111までですが、&B1111111111111111〜&B1000000000000000までは負の数で、−1から−32768に対応します。

例

&B01100100　　10進数では100
&B1010101010101010　　10進数では−21846

8進形式、16進形式、2進形式で入力された数値は、PRINTやLPRINT命令で出力すると、10進形式で表示されます。10進以外の形式で出力するときは、それぞれOCT$、HEX$、BIN$を使って文字列として出力します。

3.4.3 単精度実数型

単精度実数型の定数には、以下の3つの形式があります。

1. 6桁以内の実数。
2. 6桁以内の実数あるいは整数と、Eを使った指数表示の組み合わせ（Eの範囲はE−64からE+62)。
3. 最後に！をつけた数値（数値が6桁を越えていると、7桁目が四捨五入される)。

例

1.23
−7.06E+06
3.14!

3.4.4 倍精度実数型

倍精度実数型の定数には、以下の3つの形式があります。

1. 7桁以上の実数
2. 7桁以上の実数あるいは整数と、Eを使った指数表示の組み合わせ（Eの範囲はE−64からE+62）Eの代わりにDを使うこともできる。
3. 最後に#をつけた数値（数値が14桁を越えていると、15桁目が四捨五入される）

例

12345678
−1.5670D-12
246.876#

3.5 変数

変数は、プログラム中で使うデータを格納するものでメモリ中に用意され、英字1文字あるいは英数字2文字からなる変数名がついています。変数に値を与える場合、変数の型は、それに代入されるデータの型と一致していなければなりません。変数に値を代入する前は、数値変数は0、文字変数はヌル（文字が何も入力されていない状態）であるものと見なされます。

3.5.1 変数の型

代入するデータの型に応じた変数の型は、変数名の最後に型宣言文字をつけることによって区別されます。型宣言文字の種類は以下の4つです。

%　　整数型

!	単精度実数型
#	倍精度実数型
$	文字型

この型宣言文字を用いることにより、変数名が同じであっても別の変数として扱うことができます(例えば、A$、A! など)。なお、型宣言文字を省略すると、#(倍精度実数型)がついているとみなされます。

変数の型はプログラム中で、DEFINT・SNG・DBL・STR 命令を使用してグローバルに宣言することも可能です。この場合でも型宣言文字による指定は優先します。

3.5.2 配列変数

配列変数は、同じ性質を持つ複数個のデータの集まりで、X(1)、X(2)のように、変数名の後にカッコで番号をつけた形で用います。カッコの番号は添字といい、配列の中の何番目の要素であるかを表します。添字は常に 0 から始まります。

配列を使うには、あらかじめ DIM 命令によって配列を宣言し、変数の名前とそこに入る要素の最大数を決めます。DIM 命令を省略して配列を使用すると、添字の最大値は自動的に 10 に設定されます。

複数の添字を持つ配列変数を使うこともできます。複数の添字を使った場合、配列は添字の個数に応じた次元になり、例えば、X(0, 1, 5)のように用います。配列の次元は、いくつ添字を指定するかによって決まります。次元は、理論的には 255 次元までとれることになっていますが、実際には 1 行の長さで指定できないので、もっと少なくなります。

3.5.3 システム変数

MSX BASIC には BASIC 自身が専用の目的のために持っている変数(システム変数)があります。システム変数は以下の 9 つです。

■ TIME	1/60 秒ごとに値が増える変数で、時間を計ることができる。この変数には 0〜65535 の値を代入することもできる。ただし、割り込みが禁止されている期間中は値が変化しないので、フロッピーディスクのアクセスなどがあるときは注意が必要。
■ ERL	エラーが発生したとき、エラーが発生した行番号を値として持つ。この変数に値を代入することはできない。

- ■ ERR　エラーが発生したとき、エラーの原因を表すエラーコードを値として持つ。この変数に値を代入することはできない。
- ■ BASE(n)　画面出力に関連するテーブルのアドレスを値として持つ。
- ■ SPRITE$(n)　定義したスプライトパターンを値として持つ。
- ■ VDP(n)　VDPレジスタのデータを値として持つ。
- ■ CSRLIN　画面上のカーソルの垂直方向の位置を値として持つ。
- ■ LPOS(n)　プリンタのヘッドの水平位置を値として持つ。
- ■ POS(n)　画面上のカーソルの水平方向の位置を値として持つ。

以下に、各変数が使用するメモリの大きさを示します。各変数の領域がメモリ上に確保されるのは、変数に値を代入した時点、あるいはDIM命令で配列を宣言したときだけです。

1. 変数

- ■整数型　5バイト
- ■単精度実数型　7バイト
- ■倍精度実数型　11バイト
- ■文字型　6＋(文字数)バイト

2. 配列変数

- ■整数型　5＋2×(要素数)＋2×(次元数)＋1バイト
- ■単精度実数型　5＋4×(要素数)＋2×(次元数)＋1バイト
- ■倍精度実数型　5＋8×(要素数)＋2×(次元数)＋1バイト
- ■文字型　5＋3×(要素数)＋2×(次元数)＋1＋(各要素に含まれる文字列の総文字数)バイト

3.6 式と演算

式とは、演算（計算）の手順を示すものであり、定数や変数を演算子（計算に使う特殊記号のこと）で結んだものです。式の演算結果は1個の数値または文字列になり、単に数値や文字だけのものも式と呼びます。

例

```
"BASIC"
```

10+3/5
2
A+B/C−D

MSX BASICの演算は、次の5つに分けられます。

- ■算術演算
- ■関係演算
- ■論理演算
- ■関数
- ■文字列演算

以下にこれらの演算について説明します。

3.6.1 算術演算

算術演算子には次のようなものがあります。なお、算術式の中に文字定数や文字変数が入ってはいけません。

表 2.4 算術演算

演算子	演　算	例
^	指数演算	X^Y
−	マイナス符号	−X
*、/	かけ算、わり算	X*Y、X/Y
+、−	足し算、引き算	X+Y、X−Y

演算の優先順序も、この順番になります。順序を変更したいときにはカッコ（()）を使えば、カッコ内の演算子は他の演算子より先に実行されます。

1. ¥とMOD

整数のわり算は/のかわりに¥によって行えば、割り切れなかった場合に答の小数点以下が切り捨てられます。扱う数値が実数の場合は、演算が実行される前に小数点以下が切り捨てられます。また、余りを求めたいときにはMODを用います。この場合も、扱う数値が実数の場合は、演算が実行される前に小数点以下が切り捨てられます。

例

PRINT 10¥3

表示 3　　　　(10/3=3　余り 1)

PRINT 10¥2.5

表示 5　　　　(10/2=5　余り 0)

PRINT 13.3 MOD 4

表示 1　　　　(13/4=3　余り 1)

2. 0でのわり算

数値を 0 で割ったときには、「Division by zero」というエラーメッセージが表示され、プログラムを中断します。0 に対して負のべき乗を行った場合も同様です。

3. オーバーフロー(桁あふれ)

代入や演算の結果が、その変数の型の中で扱える範囲を越えた場合、桁あふれがおこります。この場合、「Overflow」というエラーメッセージが表示され、プログラムを中断します。

3.6.2 関係演算

関係演算子は、2 つの数値または 2 つの文字列を比較するときに用います。関係演算の結果は、真(−1)、偽(0)で出され、IF 命令などでプログラムの流れを変えるのに用いられます。以下に、関係演算子の意味と例をあげます。

表 2.5　関係演算

関係演算子	内　容	例
=	等しい	X=Y
<>、><	等しくない	X<>Y、X><Y
<	小さい	X<Y
>	大きい	X>Y
<=、=<	小さいか等しい	X<=Y、X=<Y
>=、=>	大きいか等しい	X>=Y、X=>Y

例

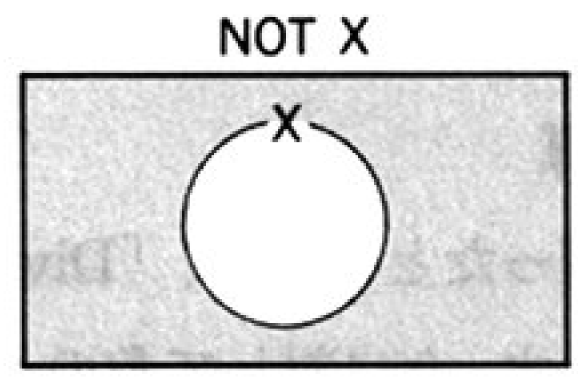
IF X=0 THEN 100
(もし X が 0 だったら、100 行にジャンプせよ)
IF A+B<>0 THEN X=A+B
(もし A+B が 0 でなかったら、X に A+B の値を代入せよ)

3.6.3 論理演算

論理演算子は複数の条件を調べたり、ビット操作やブール演算を行ったりするのに用います。論理演算の結果は、ビットごとに0または1で返されます。各論理演算の内容を以下に示します。

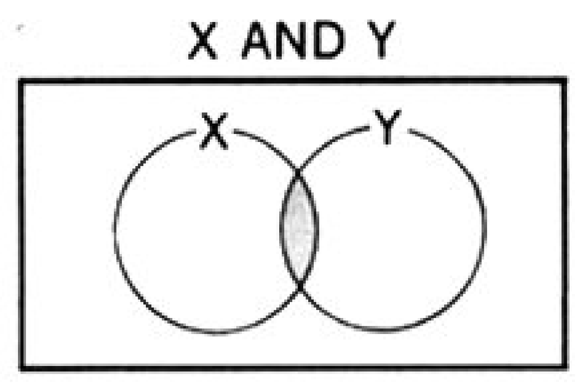
NOT（否定）

X	NOT X
1	0
0	1



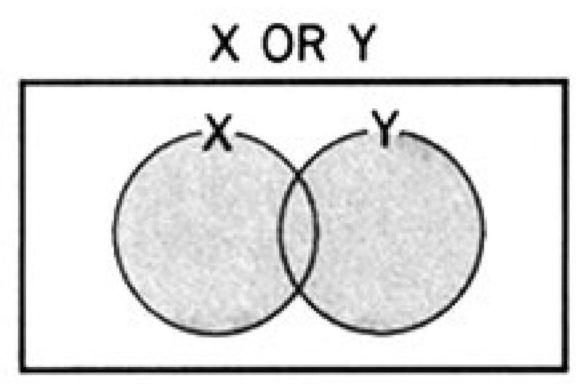
AND（論理積）

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0



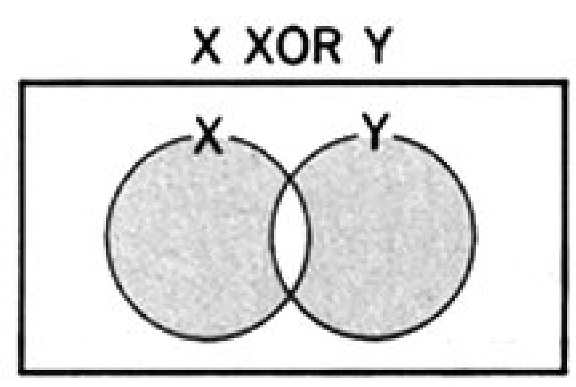
OR（論理和）

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0



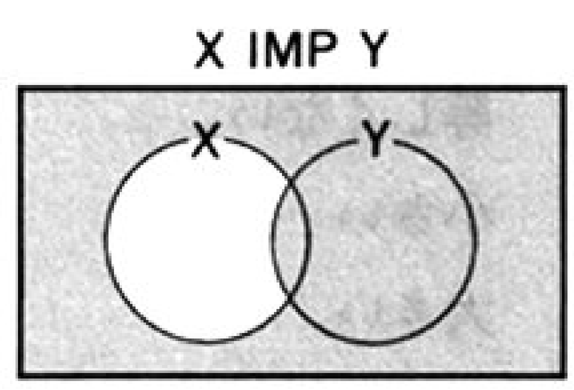
XOR（排他的論理和）

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0



IMP（包含）

X	Y	X IMP Y
1	1	1
1	0	0
0	1	1
0	0	1



EQV（同値）

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

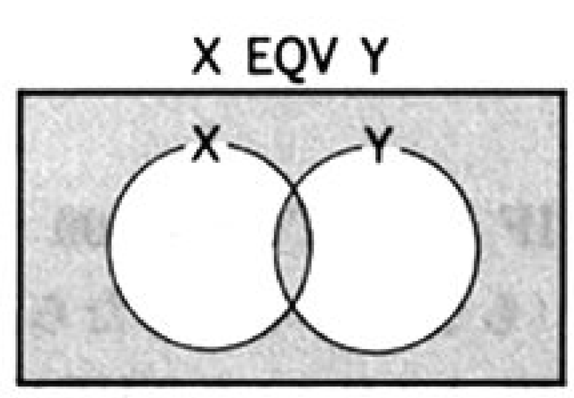


図2.3 論理演算

IF命令では、これらの論理演算子を使って、複数の条件を判断することができます。

例

IF X<0 OR 99<X THEN 100
(Xが負または99より大きければ、100行にジャンプする)
IF 0<X AND X<100 THEN X=0
(Xが正でしかも100より小さければ、Xに0を代入する)
IF NOT(A=0) THEN 20
(Aが0でなければ、20行へジャンプする)

3.6.4 関数

関数とは、指定された数値に対して、ある決まった演算を行うもので、その演算の結果を返すものです。

MSX BASICには、関数として、SINやSQRなどの数値を扱うものや、CHR$やLEFT$などの文字列を扱うものなどがあります。また、最初から定義されたものではなく、ユーザーが自由に定義できる関数(DEFFN、DEFUSR)もあります。

3.6.5 文字列演算

1. 文字列の連結

文字列は、演算子「+」を使ってつなげることができます。

例

A$="ABC": B$="DEF": C$=A$+B$: ? C$
表示 ABCDEF

2. 文字列の比較

文字列も、数値の比較に使われるものと全く同じ関係演算子(=、<、>、<>、><、<=、=<、=>、>=)を用いて比較することができます。

文字列の場合、それぞれの文字列の先頭から1文字ずつ文字を比較します。両者が全く同じ文字列の場合は、その2つの文字列は等しくなりますが、1文字でも違った場合は、その文字のキャラクタコードの大きい方の文字列が大きくなります。また、文字列の一方が短くて比較が途中で

終わった場合は、短い文字列の方が小さくなります。

例

”AB”＜”BC”
”BC”＞”AAA”
”CAT”＜”CATS”
”PEN ”＞”PEN”
”cm”＞”CM”
”BASIC”=”BASIC”

文字列を比較することによって、文字列の内容を調べたり、文字をアルファベット順に並べかえたりすることができます。

3.6.6 演算の優先順位

演算は次の順位によって行われます。

1. カッコで囲まれたもの
2. 関数
3. 指数（べき乗）
4. マイナス符号（－）
5. ＊、／
6. ￥
7. MOD
8. ＋、－
9. 関係演算子（=、＜、＞など）
10. NOT
11. AND
12. OR
13. XOR
14. IMP
15. EQV

3.7 エラーメッセージ

プログラムやダイレクトモードでの命令を実行しているときに、エラーが発生した場合、実行は中断され、エラーメッセージが画面に表示されます。

エラーメッセージは、ダイレクトモードでは、

XX・・・・・

プログラムモードでは、

XX・・・・・in YYYY

という形で表示されます。このとき XX・・・・・はエラーメッセージで、YYYY はエラーが見つかった行番号です（必ずしもエラーの原因となる行番号ではありませんので注意してください)。エラーメッセージの内容については、「3.14 エラーメッセージ一覧」を参照してください。

3.8 画面モード

MSX の画面は次のものがあります。

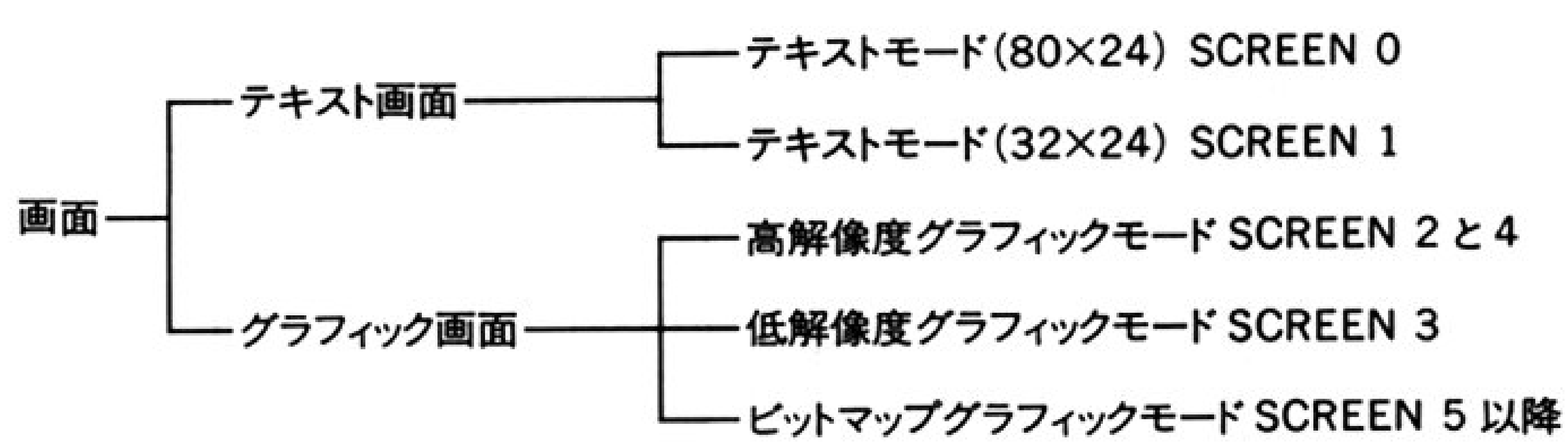


図 2.4　画面モード

テキスト画面は主に文字や数を表示するための画面で、リストやメッセージを表示することができます。テキスト画面には最大横 80×縦 24 文字表示できるモードと、最大横 32×縦 24 文字表示できるモードとがあります。

グラフィック画面は点や線を描くための画面で、細かい点や線が描ける高解像度グラフィックモードと、4×4 ドットのブロック単位で絵を描く低解像度グラフィックモードと、画面の 1 ドッ

ト毎に色を付けることができるビットマップグラフィックモードがあります。

MSX BASICではこれらの12の画面のどれを使うかを、SCREEN命令を使って指定します。BASIC起動時には、自動的に32×24文字のテキストモードが選択されますが、MSX BASIC ver. 2.0以降ではどのモードで立ち上がるかをSET SCREEN命令でバッテリーバックアップされたメモリに記憶させておくことができます。各画面の色はCOLOR命令を使って変えることができます。使用できる画面はMSX BASIC ver. 1.0では0〜3(ただしSCREEN 0は横40×縦24キャラクタ)、ver. 2.0では0〜8、ver. 3.0では0〜12（ただし9は使用できない）です。

1. テキスト画面

文字を表示するための画面です。1行の文字数はWIDTH命令を使って変えることができます。ファンクションキーを表示しているときは、画面の一番下の行は使えません(KEY ON / OFF参照)。

LOCATE命令での位置指定は、横（X軸方向）が0〜79（32×24の画面では、0〜31）、縦（Y軸方向）が0〜23の値で指定できます。

例

```
LOCATE 10,10
```

■テキストモード（縦80×横24）

最大80×24文字を表示できます。1文字の大きさは、横6×縦8ドットです。この画面では、ひらがなの右側が欠けて表示されます。WIDTH 40以下のときは文字の大きさが2倍になります。

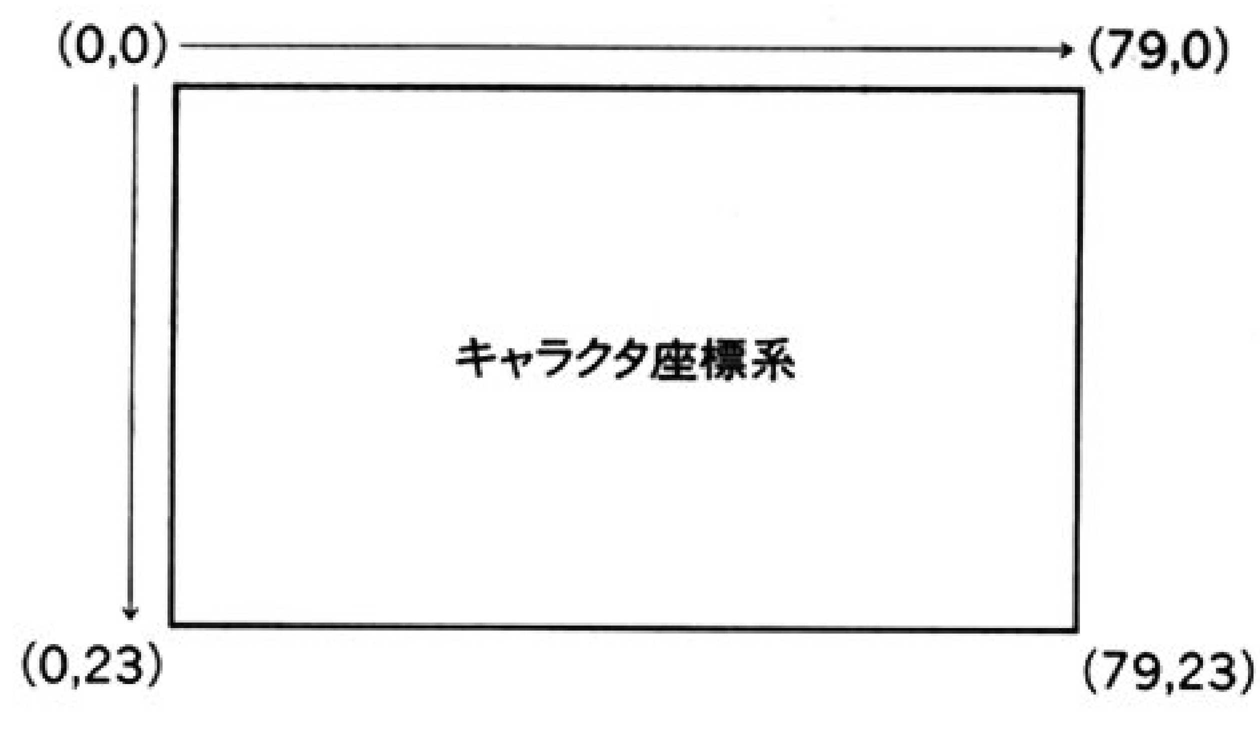


図2.5 テキストモード（80×24）

■テキストモード（縦 32×横 24）

最大 32×24 文字を表示できます。1 文字の大きさは、横 8×縦 8 ドットです。この画面では、スプライト機能を使うことができます。

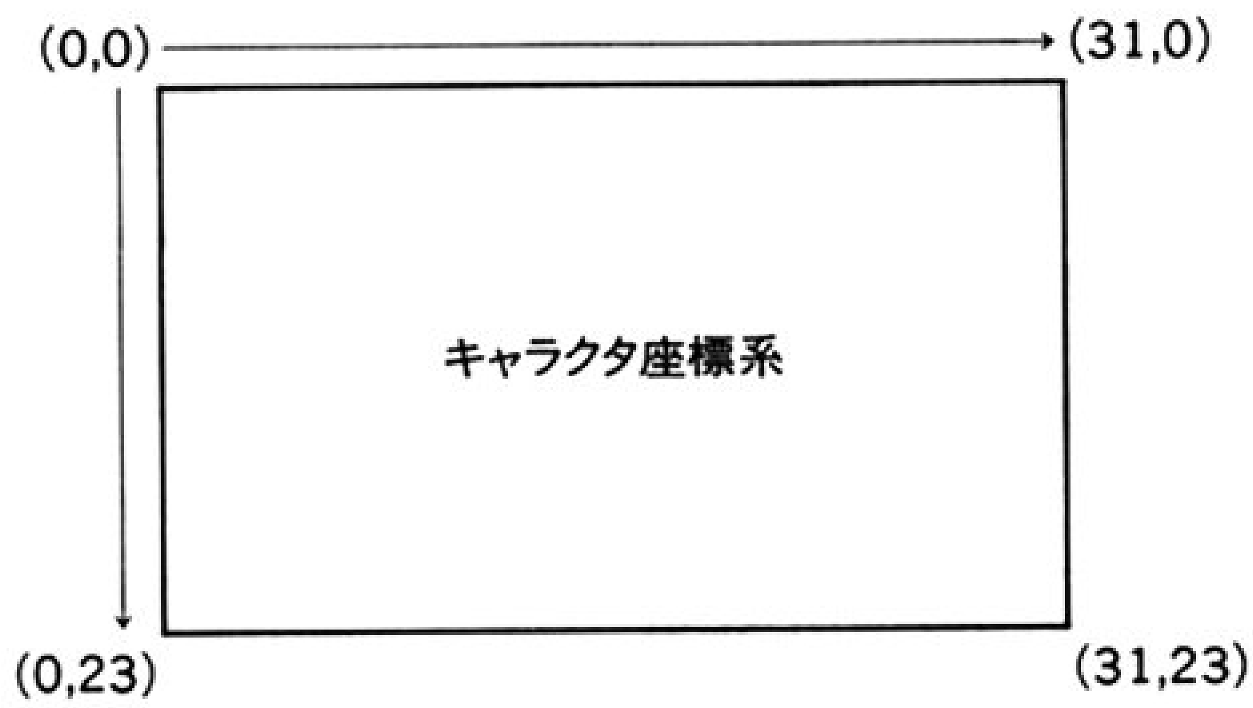


図 2.6　テキストモード（32×24）

2. グラフィック画面

点や線、絵を描くための画面です。スプライト機能を使うこともできます。グラフィック画面はプログラム中でだけ使うことができます。プログラムの実行が終ると、画面は自動的にテキスト画面に戻ります（SCREEN 参照）。

グラフィック画面で文字を表示するには、OPEN 命令で"GRP:"を指定してから、PRINT #命令を使って行いますが漢字グラフィックモードでは普通の PRINT 文でも可能です。

グラフィック画面でどの位置に点を表示するかなどの指定は、横（X 軸方向）が 0〜511、縦（Y 軸方向）が 0〜211 までの値で行います。

例

```
PSET(100,100)
```

■高解像度グラフィックモード（SCREEN 2 と 4）

横 256 個×縦 192 個のドットを使うことができる画面です。細かい点や線を描くことができます。横方向の 8 ドットごとに、2 色までの色を使えます。

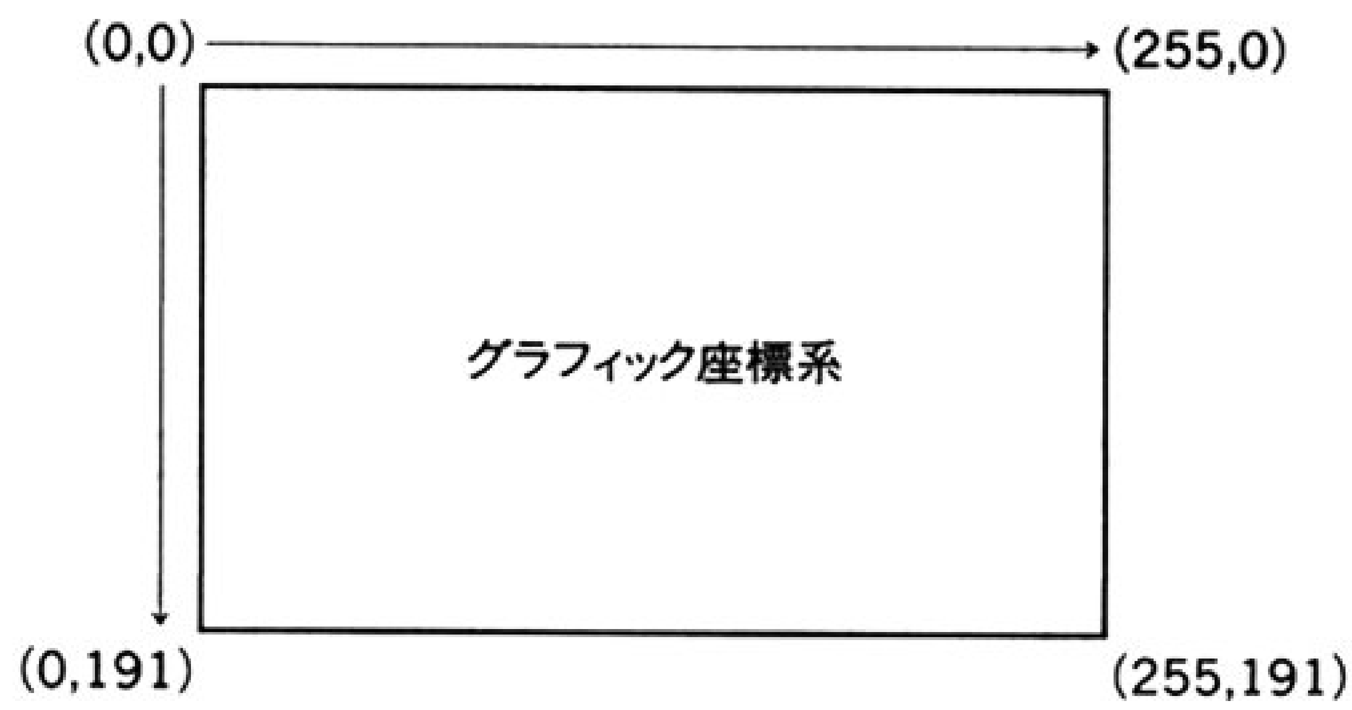


図 2.7 高解像度グラフィックモード

■低解像度グラフィックモード (SCREEN 3)

横 64×縦 48 ブロックのグラフィック画面で、点を描く単位が 4×4 ドットになっています。この 1 ブロックごとに色を付けることができます。

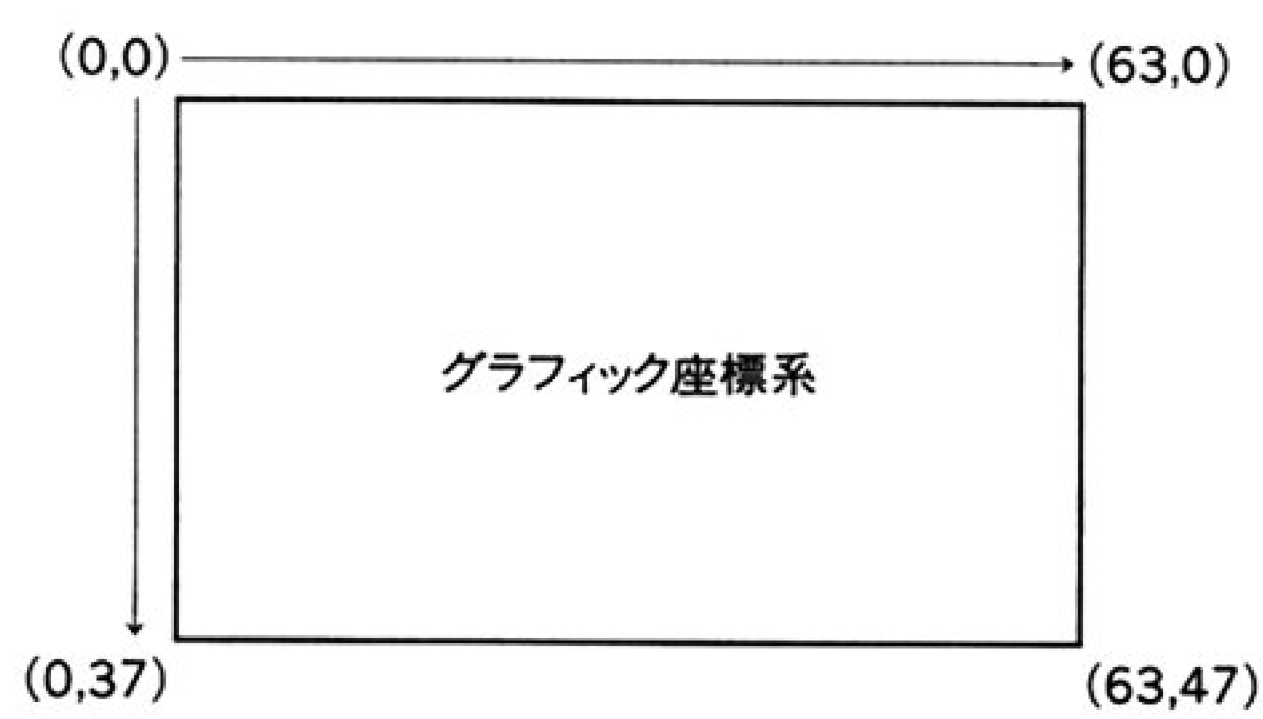


図 2.8 低解像度グラフィックモード

■ビットマップグラフィックモード (SCREEN 5 以降)

横 256 個(SCREEN 6 と 7 では 512)×縦 212 個のドットを使うことができる画面です。細かい点や線を描くことができます。1 ドット毎に 16 のパレットから選んだ色が使えます。ただし SCREEN 8 では各ドットに 256 の固定色から選んだ色を使います。

3.9 カラー番号

MSX BASIC では、基本的には次の 16 色の色を使うようになっています。色の指定は、それぞれのカラー番号で行います。指定の方法は、それぞれの命令を参照してください。

表 2.6　カラー番号

カラー番号	色	カラー番号	色
0	透明（周辺色と同じ色）	8	赤
1	黒	9	明るい赤
2	緑	10	黄
3	明るい緑	11	明るい黄
4	暗い青	12	暗い緑
5	明るい青	13	紫
6	暗い赤	14	灰
7	水色	15	白

各カラー番号がこのように決っているとき、「カラーパレットが初期状態にある」と言います。カラーパレットを上記以外の色にするときは、COLOR =を使ってパレットを変更します。例えば、

COLOR =(8,7,7,0)

とすることによってパレット 8 は、明るい黄色になります。SCREEN 6 では、0〜3、それ以外では 0〜15 のパレットが使えます。ただし SCREEN 8 では、パレットは使いません。カラー番号は次のように固定した色を意味しています。

&B00000000	カラー番号を 2 進数で表したもの
GGGRRRBB	対応する緑、赤、青の明るさ

例えば、&B1110000 なら明るい緑になります。

3.10 スプライト機能

MSX BASIC には、解説した 4 つの画面のほかにスプライト面というものがあります。スプライト面は、0 番から 31 番までの 32 枚があり、1 枚の面に 1 個のスプライトパターンを表示することができます。

スプライト面は、普通の画面の前に重ねてある面と考えることができます。スプライト面にパターンを表示すると、そのパターンが元の画面の絵や文字の上に重なって見えます。また、2 つ以上のスプライトパターンを重ねると、スプライト面番号の小さいほうのパターンが手前に見え、立体的なパターンのようになります。

スプライト機能は、SCREEN 0 以外のモードを使っているときに使うことができます。

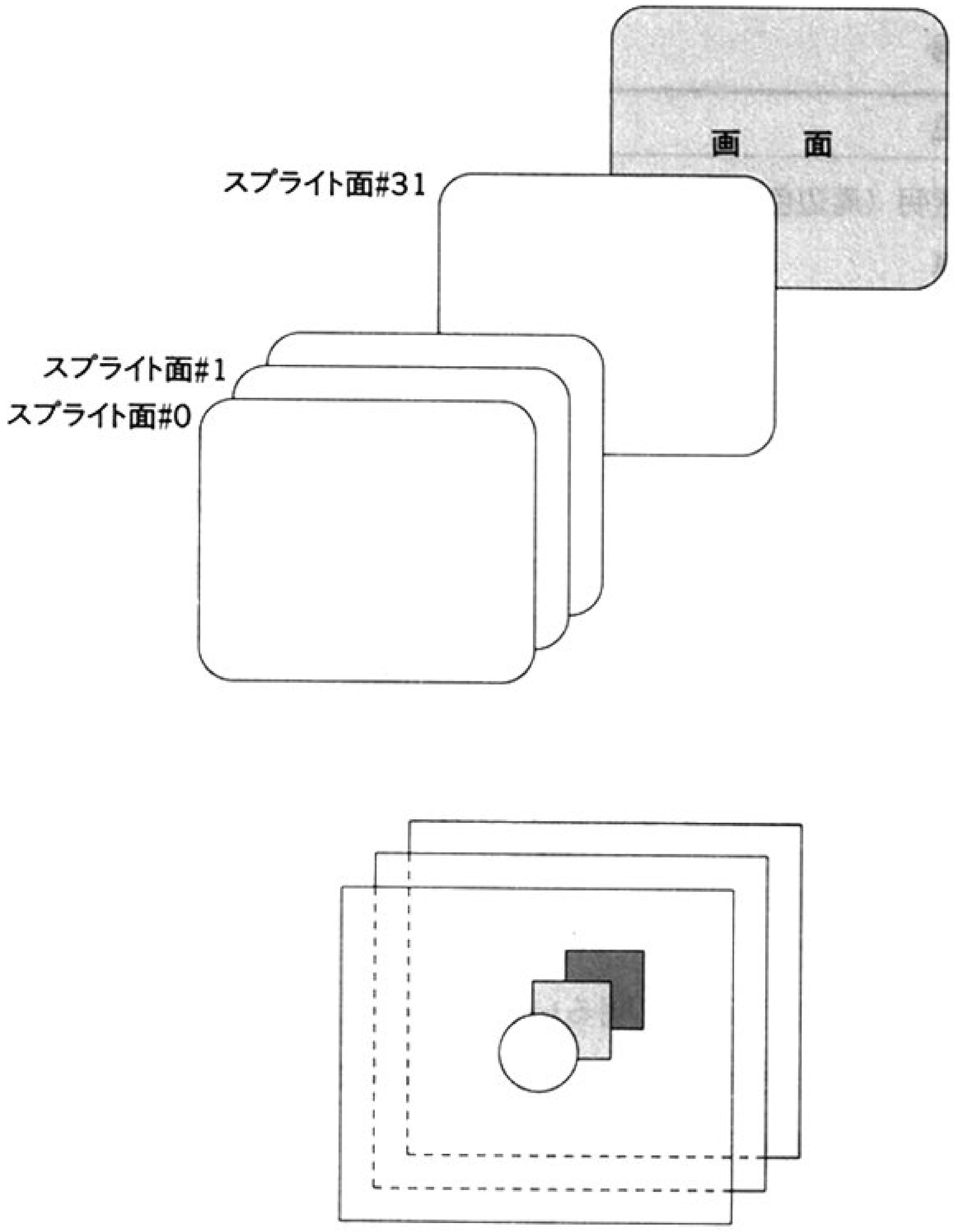


図 2.9 スプライト

パターンの大きさは、8×8 ドット、あるいは 16×16 ドットです。8×8 のときは、256 個、16×16 のときは 64 個までのパターンを定義しておくことができます。また、パターンは同時に 32 個まで表示できますが、横方向に 5 個（SCREEN 4 以降では 9 個）以上重なると、スプライト番号の小さいパターン 4 つ（SCREEN 4 以降では 8 つ）だけが表示されます。

スプライトを表示するには、以下の手順で行います。

1. SCREEN でスプライトパターンの大きさを決める。
2. SPRITE$でスプライトパターンを定義する。
3. SCREEN 4 以降であれば COLOR SPRITE で各プレーンの色を定義する。
4. PUT SPRITE で、スプライト面にパターンを表示する。

このほかに、スプライトに関する命令として、以下のものがあります。

ON SPRITE GOSUB	パターンが重なったときに、指定した行にジャンプする。
SPRITE ON/OFF/STOP	ON SPRITE GOSUB で指定された行へジャンプするかどうかを決める。

3.11 ファイル

ファイルとは、意味を持つ情報の集まりです。カセットテープに保存したプログラムやデータは、ファイルとして扱われます。また他の周辺装置への出力も、まとまったデータを送るという意味で、ファイルというとらえかたで扱うことができます。

1. ファイル番号

プログラム中でファイルを扱うときには、まずOPEN命令でファイルの使用開始を宣言し、ファイルに対して番号を割り当て、また、番号と対応した入出力用の領域を確保しなければなりません。この番号をファイル番号と呼びます。その後のファイルへの入出力は、すべてこのファイル番号によって指定されたファイルに行われます。

2. ファイルを扱う場所

ファイルを操作する場所（ファイルの入出力の対象となる場所）には、以下の5つがあります。

ファイルを扱う所	指定の方法	なされる操作
カセットレコーダ	"CAS:ファイル名"	入出力を行う
メモリディスク	"MEM:ファイル名"	入出力を行う
テキスト画面	"CRT:"	出力のみ
グラフィック画面	"GRP:"	出力のみ
プリンタ	"LPT:"	出力のみ

3. ファイル名

ファイル名とは、ファイルにつける名前のことです。ファイル名を必要とするのは、カセットテープやメモリディスクへの入出力を行う場合で、その他の場合は省略できます。カセットテープで使えるファイル名は、6文字で、メモリディスクでは8文字です。6（または8）文字より小さい場合は、あいた部分には空白があてはめられます。6（または8）文字より大きい場合は、先頭から6（または8）文字がファイル名とみなされ、それ以降の文字は無視されます。また、ファイル名の中に、コロン（：）を用いてはいけません。

3.12 割り込み

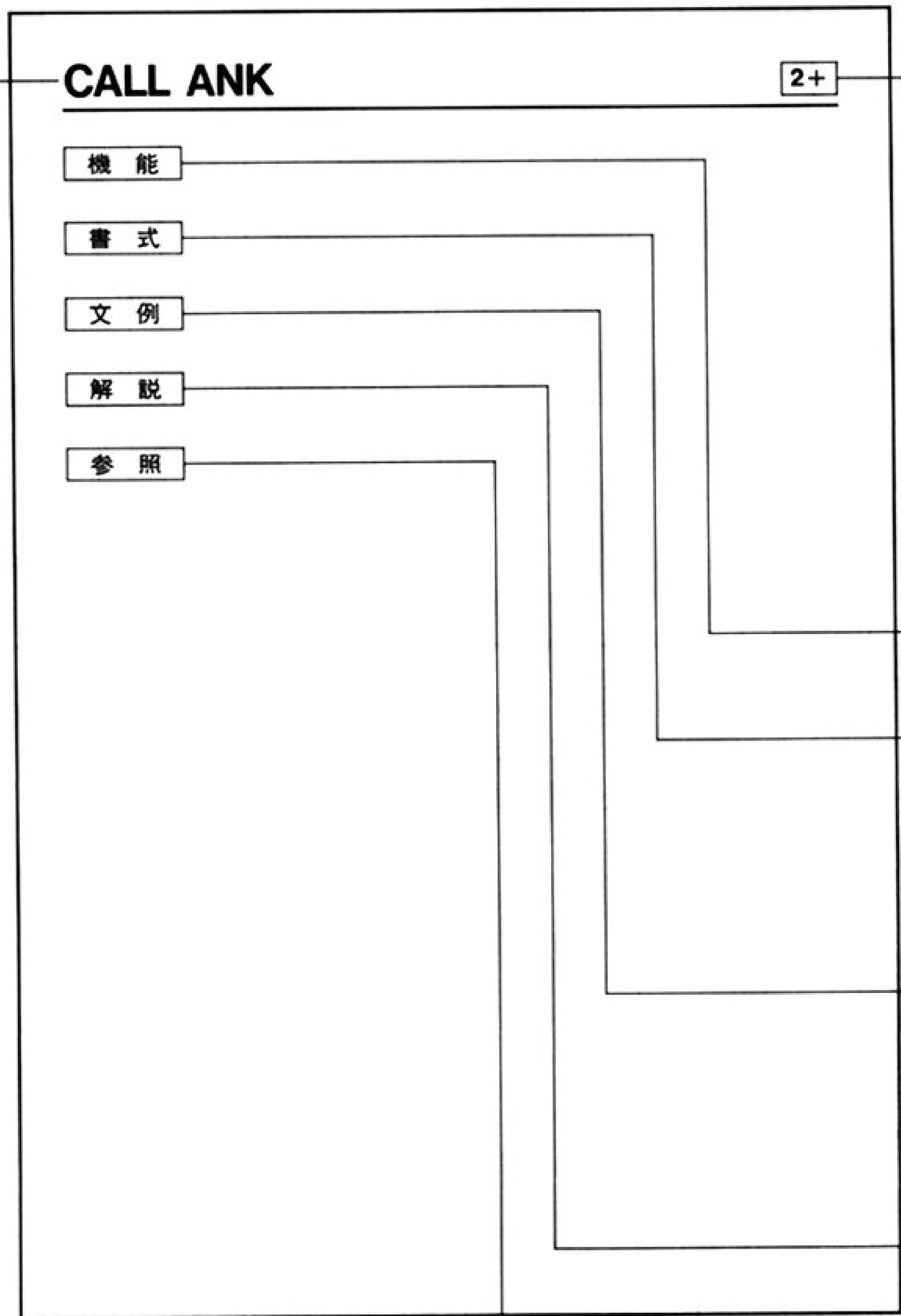
プログラムの実行中に、何か特別な事が起きたとき、現在実行中の処理を一時中止して、その特別な事柄に対応した処理を先に実行させることを、割り込みといいます。

例えば、ファンクションキーが押されたときに対応する処理を実行するようにあらかじめ指定しておくと、プログラムの実行中にファンクションキーが押された時点で割り込みがかかり、そのとき実行していた処理を中断して指定された処理を優先的に実行したあと、中断されていた処理が再開されます。

以下に、MSX BASICが用意している割り込みを、優先順に示します。

1.	ファンクションキー割り込み	ON KEY GOSUB
2.	ストップキー割り込み	ON STOP GOSUB
3.	スプライトの衝突割り込み	ON SPRITE GOSUB
4.	ジョイスティックのトリガ割り込み	ON STRIG GOSUB
5.	インターバルタイマ割り込み	ON INTERVAL GOSUB

3.13 ステートメント



ステートメント名
命令の名前です。

対応機種
その命令が使える機種を表す記号です。それぞれの記号には次のような意味があります。

無印	すべてのMSXで使用可能
[2]	MSX2、MSX2＋で使用可能
[2＋]	MSX2＋で使用可能
[Disk]	ディスクドライブを搭載した機種で使用可能

命令の内容を簡単に説明しています。

命令の書き方を示しています。＜＞で囲まれた項目は、何らかの数値や文字を指定し、[]で囲まれた項目は、省略可能であることを示します。

実際に命令を使った例（サンプルプログラム）があげてあります。[表示]で示された数値および文字列は、そのプログラムの実行結果です。

サンプルプログラムに引き続いて、その命令についてのくわしい説明と、使用するときの注意などが述べられています。

参照する命令や資料などをあげてあります。

ABS

機　能

<数式>の絶対値を返します。

書　式

ABS(<数式>)

文　例

```
PRINT ABS(-1.2)
```

表示 1.2

ASC

機　能

<文字式>のキャラクタコードを返します。

書　式

ASC(<文字式>)

文　例

```
PRINT ASC("a")
```

表示 97

解　説

指定した<文字式>がいくつかの文字からなる文字列の場合は、一番最初の文字のキャラクタコードが返されます。文字とキャラクタコードの対応については、キャラクタコード表を参照してください。

参　照

CHR$、キャラクタコード表

ATN

機　能

<数式>の逆正接（アークタンジェント）を計算します。

書　式

ATN(<数式>)

文　例

```
PRINT ATN(1) * 4
```

表示 3.1415926535898

AUTO

機　能

プログラム行の先頭に行番号を自動的に発生させます。

書　式

AUTO[<行番号>][,<増分>]

文　例

```
AUTO 1000,100
```

解　説

最初の行番号も番号の増加分も省略できますが、この場合は自動的に10が採用されます。
例えば、

AUTO

行番号は 10 を先頭に 20、30・・・・・と付けられます。

AUTO 100

行番号は 100 を先頭に 110、120・・・・・と付けられます。

AUTO ,5

行番号は 10 を先頭に 15、20・・・・・と付けられます。

プログラム中に既に存在する行と同じ行番号が発生した場合には、行番号の後にアスタリスク（*）が表示されます。このとき、何も入力しないでリターンキーを押せば、その行は前の内容のままです。

AUTO機能を中止したいときは、CTRL + C キーまたは CTRL + STOP キーを押します。

BASE

機能

画面出力に関連するVRAM上のテーブルのアドレスを返します。

書式

BASE(<数式>)

文例

PRINT BASE(10)

解説

カッコ内の数値で指定されたVRAM上のテーブルのアドレスを返します。MSX1では、VDPとしてTMS9918、MSX2ではV9938、MSX2+ではV9958を使っています。

参照

VDP

BEEP

機能

スピーカを鳴らします。

書式

BEEP

解説

SET BEEP命令で設定した音色と大きさで、スピーカーを鳴らします。

次のように、PRINT命令でCHR$(7)を実行しても、BEEPと同じ結果が得られます。

PRINT CHR$(7)

参 照

コントロールコード表

BIN$

機 能

<数式>を 2 進数の文字列に変換します。

書 式

BIN$(<数式>)

文 例

```
PRINT BIN$(50)
```

表示 110010

参 照

VAL、OCT$、HEX$

BLOAD

機 能

機械語プログラム、または画面データファイルを読み込みます（ロードする)。

書 式

BLOAD<ファイルスペック>[[,R]] | [,S]][,<オフセット>]

文 例

```
BLOAD "A: DEMO",R
```

解 説

<ファイルスペック>で指定した機械語プログラムファイルをメモリ上にロードします。カセットファイルに対してはファイル名を省略することができますが、ディスクファイルに対しては省略することはできません。<ファイルスペック>の中で<デバイス名>を省略すると現在選択されているディスクドライブ（カレントドライブ）とみなさ

れます。カセットファイルを対象とする場合は"CAS:"とします。
<オフセット>を省略すると機械語プログラムはBSAVEでセーブするときに指定した<開始アドレス>からロードしますが、<オフセット>を指定した場合には、セーブの際に指定された開始アドレスに、<オフセット>を加えた番地からロードします。したがって、<オフセット>を付けてロードするプログラムは、リロケータブルな性質をもっていなければなりません。
Rオプションを指定すると、プログラムをロード後、BSAVEで指定しておいた実行開始アドレスから、ただちに実行されます。このとき、既に開かれているファイルがあればファイルは開かれたまま機械語プログラムが実行されます。
Sオプションを指定すると、ディスクファイルの内容がVRAMにロードされます。<オフセット>を省略する場合にはカンマ(,)も省略してください。

参　照

BSAVE

BSAVE

機　能

機械語プログラム、または画面データを保存(セーブ)します。

書　式

BSAVE<ファイルスペック>,<開始アドレス>,<終了アドレス>[[,<実行開始アドレス>|[,S]]

文　例

```
BSAVE "B: DEMO",&HA100,&HA2FF
```

解　説

<開始アドレス>から<終了アドレス>までのメモリ上に置かれている機械語プログラムを、<ファイルスペック>で指定したファイルにセーブします。
<ファイルスペック>の中の<デバイス名>を省略した場合、カレントドライブにセーブされます。カセットファイルの場合は"CAS:"としてください。
<実行開始アドレス>を指定しておくと、BLOADでRオプションを指定して、プログラムのロード後に自動的に<実行開始アドレス>から実行させることができます。
Sオプションを指定すると、画面データをディスクファイルとしてセーブすることができます。

CALL

機　能

拡張ステートメントを呼び出します。

書　式

CALL＜拡張ステートメント名＞[(＜引数＞[,＜引数＞・・・])]

解　説

拡張 ROM カートリッジなどで供給される拡張ステートメントを呼び出します。CALL は_（アンダースコア）で代用することができます。

CALL AKCNV　　2+

機　能

＜文字列＞中の全ての文字を全角文字に変換して＜文字変数＞に代入します。

書　式

CALL AKCNV(＜文字変数＞,＜文字列＞)

文　例

```
10 CALL AKCNV(A$,"ABC 漢字ｲﾛﾊ")
20 PRINT A$
```

表示　ＡＢＣ漢字イロハ

解　説

半角文字（カタカナ、英数字等）を全角の文字に変換します。もともと全角の文字はそのままです。

CALL ANK　　2+

機　能

漢字ドライバを終了します。

書　式

CALL ANK

文　例

```
CALL ANK
```

解　説

漢字モードを終了し、ANKモードとなります。ただし、漢字ドライバ用に確保されたメモリは解放されません。

CALL CLS

2+

機　能

漢字モードの画面をクリアします。

書　式

CALL CLS

文　例

```
CALL CLS
```

解　説

漢字モードでは通常のCLSは使えません。漢字モード以外でもCALL CLSは有効です。

CALL FORMAT

Disk

機　能

フロッピーディスクをフォーマット（初期化）します。

書　式

CALL FORMAT

文 例

CALL FORMAT

解 説

Disk BASIC や MSX-DOS で使用できるように、ディスクを初期化します。実行するとドライブ名をきいてくるので、指定します。間違って既に使用しているディスクをフォーマットしてしまうと、その中のデータは完全に失われてしまいます。十分注意してください。

CALL JIS

2+

機 能

<文字列>の最初の2バイトを16進4桁のJISコードに変換して<文字変数>に代入します。

書 式

CALL JIS(<文字変数>, <文字列>)

文 例

```
10 CALL JIS(A$,"漢字")
20 PRINT A$
```

表示 3441

CALL KACNV

2+

機 能

<文字列>中の半角文字に変換できる文字を全て半角文字に変換して<文字変数>に代入します。

書 式

CALL KACNV(<文字変数>, <文字列>)

文 例

```
10 CALL KACNV(A$,"10人のプログラマー")
```

```
20 PRINT A$
```
表示 10 人ノプ゛ロク゛ラマー

CALL KANJI

2+

機能

漢字ドライバを起動します。

書式

CALL KANJI[n]

文例

CALL KANJI3

解説

nは、0、1、2、3のうちどれか1文字で、漢字フォントとSCREENのインタレースモードを指定します。省略すると0と見なされます。

モード	文字の大きさ (全角X×Y)	表示文字数 (最大横×縦)
KANJI0	16×16	32×13
KANJI1	12×16	40×13
KANJI2	16×16	32×24
KANJI3	12×16	40×24

CALL KANJI(またはCALL KANJI0)はフォントをMSXの標準漢字ROMから取って表示します。

CALL KANJI1はフォントをMSXの標準漢字ROMからとり、横16ドットを12ドットに圧縮して表示します。ただし、松下電器の仕様による12ドットフォントROMがシステムに存在する場合はこれを表示します。

CALL KANJI2はCALL KANJI(またはCALL KANJI0)と同じですが、インタレースモードが使用され縦方向の表示文字数が増加します。

CALL KANJI3はCALL KANJI1と同じですが、インタレースモードが使用され縦方向の表示文字数が増加します。

MSX標準の日本語フロントエンドプロセッサ(以下、MSX-JE)があれば、このステー

トメントの実行時またはシステムの立ち上げ時に組み込まれます。直接入力モード(ANK)と間接入力モード(漢字)は、[CTRL]+[SPACE]もしくは[GRAPH]+[SELECT]によって切り換えられます。MSX-JEが存在しないときは、単漢字変換機能が使用できます。

MSX起動後の一番最初のCALL KANJIの実行時には注意が必要です。その時点でのHIMEM(CLEAR文による)の設定はキャンセルされ、全ての変数およびソフトウェアスタック(FOR/NEXT、GOSUB/RETURN用)がクリアされます。これは漢字ドライバのワークエリア(もしあればMSX-JEも)を確保する処理が行われるからです。2回目以降の実行は問題なく行われます。

CALL KANJI命令などを実行し、漢字表示ができる状態を「漢字モード」と呼びます。さらに漢字モードでかつSCREEN文で0もしくは1が設定されている状態を「漢字テキストモード」と呼び、漢字モードでかつSCREEN文で2以上の画面モードが設定されている状態を「漢字グラフィックモード」と呼びます。

BASICのコマンド待ちの状態では漢字テキストモードになっており、漢字の入出力が可能です。漢字グラフィックモードではプログラムによる漢字の出力のみが可能です。

漢字モードにおけるWIDTH

■漢字テキストモード

CALL KANJI実行後のWIDTHはANKモードでのWIDTHをもとに、以下のように決定されます。CALL KANJI実行後に直接SCREEN文やWIDTH文で指定することもできます。

1. ANKモードでSCREEN 0を使用していた場合。

■ KANJI0かKANJI2

ANKモードでのWIDTHの値の4/5が漢字テキストモードでのWIDTHになります。

■ KANJI1かKANJI3

ANKモードでのWIDTHの値がそのまま漢字テキストモードでのWIDTHになります。

2. ANKモードでSCREEN 1を使用していた場合。

■ KANJI0かKANJI2

ANKモードでのWIDTHの値がそのまま漢字テキストモードでのWIDTHになります。

■ KANJI1かKANJI3

ANKモードでのWIDTHの値の5/4が漢字テキストモードでのWIDTHになります。

■漢字グラフィックモード

常に表示できる最大に設定されます。

WIDTH による SCREEN モードの選択

- KANJI0 か KANJI2

  WIDTH が 26 から 32 の時は 256 ドットモード（VDP の GRAPHIC 4 モード）が、33 から 64 の時は 512 ドットモード（VDP の GRAPHIC 6 モード）が選択されます。
- KANJI1 か KANJI3

  WIDTH が 26 から 40 の時は 256 ドットモード（VDP の GRAPHIC 4 モード）が、41 から 80 の時は 512 ドットモード（VDP の GRAPHIC 6）が選択されます。

CALL KEXT　2+

機　能

＜機能＞が 0 なら＜文字列＞中の半角文字だけを、1 なら全角文字だけを抜き出して＜文字変数＞に代入します。

書　式

CALL KEXT(＜文字変数＞,＜文字列＞,＜機能＞)

文　例

```
10 CALL KEXT(A$,"今日ﾊ良ｲ天気ﾃﾞｽ",0)
20 PRINT A$
```

表示　ﾊｲﾃﾞｽ

CALL KINSTR　2+

機　能

漢字を含む文字列を検索します。

書　式

CALL KINSTR(＜数値変数＞[,＜数式＞],＜文字列1＞,＜文字列2＞)

文　例

```
10 CALL KINSTR(A,"A亜B","B")
```

```
20 PRINT A
```

表示 3

解説

＜文字列1＞の中から＜文字列2＞を探しだし、見つかればその位置を、見つからなければ0を＜数値変数＞に代入します。＜数式＞は探し始める位置を文字数を単位として指定するもので、省略した場合は1とみなされます。

CALL KLEN

2+

機能

漢字を含む文字列の文字数を返します。

書式

CALL KLEN(＜数値変数＞,＜文字列＞[,＜機能＞])

文例

```
10 CALL KLEN(A,"今日ﾊ良ｲ天気です")
20 PRINT A
```

表示 9

解説

＜機能＞が0(もしくは省略)のときは＜文字列＞の全体の文字数を、1のときは＜文字列＞中の半角文字数を、2のときは＜文字列＞中の全角文字数を＜数値変数＞に代入します。

CALL KMID

2+

機能

漢字を含む文字列の一部を取り出します。

書式

CALL KMID(＜文字変数＞,＜文字列＞,＜数式1＞[,＜数式2＞])

文例

```
10 CALL KMID(A$,"今日はヨい天気です",3,3)
20 PRINT A$
```

表示 はヨい

解説

<文字列>中の<数式1>番目の文字から<数式2>文字分だけ抜き出して<文字変数>に代入します。<数式2>が省略された場合は<数式1>番目の文字から終りまで全ての文字が代入されます。

CALL KNJ

2+

機能

漢字コードに相当する漢字を獲得します。

書式

CALL KNJ(<文字変数>, <文字列>)

文例

```
10 CALL KNJ(A$,"3441")
20 PRINT A$
```

表示 漢

解説

<文字列>で指定される4桁の漢字コードに相当する漢字1文字を<文字変数>に代入します。漢字コードが8000H未満であればJIS、それ以上の場合はシフトJISとみなします。

CALL KTYPE

2+

機能

文字のタイプを調べます。

書　式

CALL KTYPE(<数値変数>, <文字列>, <数式>)

文　例

```
10 CALL KTYPE(T,"ｺﾚﾊ漢字",4)
20 PRINT T
```

[表示] 1

解　説

<文字列>中の<数式>番目の文字のタイプを<数値変数>に代入します。タイプは、0＝半角、1＝全角です。

CALL MEMINI　[2]

機　能

メモリディスクで使うメモリの上限を指定します。

書　式

CALL MEMINI(<メモリディスクの上限>)

文　例

```
CALL MEMINI(&H7000)
```

[表示] 27904 bytes allocated

解　説

範囲は、0～&H7FFF です。初期状態では、&H7FFF が指定されています。
CALL MEMINI 命令が実行されると、メモリディスクに割り当てられたメモリの量が表示されます。

CALL MFILES　[2]

機　能

メモリディスク中のファイル名を表示します。

書　式

CALL MFILES

文　例

```
CALL MFILES
```

CALL MKILL　2

機　能

メモリディスク中のファイルを削除します。

書　式

CALL MKILL(<ファイル名>)

文　例

```
CALL MKILL("SAMPLE")
```

CALL MNAME　2

機　能

メモリディスク中のファイル名を付け替えます。

書　式

CALL MNAME(<新ファイル名>AS<旧ファイル名>)

文　例

```
CALL MNAME("TEST" AS "TEST.AA")
```

CALL PALETTE　2+

機　能

カラーパレットの初期化または設定を行ないます。

書　式

CALL PALETTE[(＜パレット番号＞,＜赤輝度＞,＜緑輝度＞,＜青輝度＞)]

文　例

```
CALL PALETTE(15,4,4,4)
```

解　説

CALL KANJI ステートメントにより画面を漢字テキストモードにしたときに、パレットを変更するために使用します。漢字モードでないときでも使用できます。
パラメータをすべて省略した場合は、パレットを初期状態にします。パラメータが指定された場合、＜パレット番号＞で指定されたパレットを＜赤輝度＞、＜緑輝度＞、＜青輝度＞の色に設定します。なお、パレット番号や各輝度を省略することはできません。

CALL SJIS　[2+]

機　能

＜文字列＞の最初の2バイトを16進4桁のシフト JIS コードに変換して＜文字変数＞に代入します。

書　式

CALL SJIS(＜文字変数＞,＜文字列＞)

文　例

```
10 CALL SJIS(A$,"漢字")
20 PRINT A$
```

[表示] 3441

CALL SYSTEM　[Disk]

機　能

MSX-DOS を起動します。

書式

CALL SYSTEM

文例

```
CALL SYSTEM
```

解説

SYSTEM コマンドを実行すると、DOS に戻る前にすべてのファイルは閉じられ、メモリ中のプログラムやデータは失われます。

この命令が使用できるのは、MSX-DOS から Disk BASIC を起動した場合だけです。そうでない場合は、「Illegal function call」エラーになります。

CDBL

機能

<数式>の値を倍精度実数に変換します。

書式

CDBL(<数式>)

文例

```
PRINT CDBL(1/3)
```

表示 .33333333333333

CHR$

機能

<数式>のキャラクタコードを持つ文字を返します。

書式

CHR$(<数式>)

文例

```
PRINT CHR$(65)
```

表示 A

数値は 0〜255 の範囲でなければなりません。文字とキャラクタコードの対応については、キャラクタコード表を参照してください。

参　照

ASC、キャラクタコード表

CINT

機　能

＜数式＞の小数点以下を切り捨てて、実数に変換します。

書　式

CINT(＜数式＞)

文　例

```
PRINT CINT(123.45678)
```

表示 123

解　説

＜数式＞の値が－32768〜32767 の範囲外のときは、「Overflow」エラーになります。

CIRCLE

機　能

グラフィック画面に円や楕円を描きます。

書　式

CIRCLE[STEP](X,Y),＜半径＞[,＜カラーコード＞][,＜開始角度＞][,＜終了角度＞][,＜比率＞]

文　例

```
CIRCLE(128,96),50,1
```

解説

SCREEN 2以上の画面で円を描きます。中心座標は、X軸方向に255(SCREEN 6、7では511)、Y軸方向に191(SCREEN 5以降では211)ありますので、その範囲内の数値を指定できます。また、STEPを用いて中心を指定することもできます(STEPを用いた指定の方法についてはLINE命令を参照)。

色を指定すると、その色で円が描かれます。色の指定を省略すると、COLORで指定された前景色が用いられます。

■弧と扇形

```
CIRCLE(128,96),50,1,3/4*3.14,1/4*3.14
```

のように、円の開始角度(3/4*3.14)と終了角度(1/4*3.14)を指定すれば、その範囲の弧が描かれます。それぞれにマイナス記号をつけると、扇形が描かれます。開始角度、終了角度の指定は、-2πから2πの範囲内で行います。

■楕円

```
CIRCLE(128,96),50,1,,,0.8
```

のように、最後に比率(0.8)を指定すれば、楕円が描かれます。比率は「垂直方向の半径/水平方向の半径」の値で指定します。値が1より小さいと横に平べったい楕円となり、1より大きいと縦に細長い楕円となります。

参照

SCREEN、COLOR

CLEAR

機能

変数の初期化と使用するメモリ領域の大きさを指定します。

書式

CLEAR[<文字列領域の大きさ>[,<メモリの上限>]]

文例

```
CLEAR 200,&HD000
```

解説

CLEARを実行すると、すべての変数が初期化され、DEF命令(DEF FN、DEFINT/SNG/DBL/STR)の定義は無効になります。初期化されると、数値変数は0に、文字変数はヌル(何も文字が代入されていない状態)になります。

また、

CLEAR 300,&HD000

のような指定では、変数の初期化のほかに、文字変数の内容をしまっておく文字列領域の大きさ（この場合、300）と、BASICが使用するメモリの上限（この場合、&HD000）を設定することができます。文字列領域の大きさは、メモリの許す限りの値まで指定できます。省略したときは200になります。また、メモリの上限は&H831F～&HF380の範囲内で指定できます。しかし、Disk BASICなどのワークエリアを壊さないように注意しなければなりません。メモリの上限を設定するときは、文字列領域の大きさの指定を省略できません。メモリの上限については、「4.2　ユーザーエリアの詳細」を参照して下さい。

参照

FRE

CLOAD

機能

カセットテープのプログラムをロードします。

書式

CLOAD[<ファイルスペック>]

文例

CLOAD "TEST"

解説

カセットテープにCSAVEでセーブされたプログラムをロードします。

CLOADが実行されると、<ファイルスペック>で指定されたファイル名のプログラムをカセットテープ上で探します。探している最中に別のプログラムを見つけると「Skip:ファイル名」と画面に表示され、指定されたプログラムが見つかると「Found:ファイル名」と表示してロードを開始します。

CLOADの後ろに指定するファイル名を省略することもできます。省略したときは、CLOADが実行されて最初にみつけたプログラム(CSAVEでセーブされたもの)をロードします。CLOAD命令を実行すると、そのときメモリ上にあったプログラムは無くなってしまうので注意してください。また、CLOADで、SAVE命令によってセーブされた

プログラムをロードすることはできません。

参照

CSAVE、LOAD

CLOAD ?

機能

カセットテープのプログラムとメモリ上のプログラムを比較します。

書式

CLOAD?[<ファイルスペック>]

文例

CLOAD ? "TEST"

解説

メモリ上にあるプログラム(LIST命令で画面に表示されるもの)と、カセットテープにセーブされたプログラムが同じかどうか調べます。同じであれば「Ok」と表示され、違っていると「Verify error」と表示されます。CLOAD?命令は、普通CSAVE命令を実行した後に、セーブが正しく行われたかどうかを確認するために使われます。

CLOAD?の後ろに指定するファイル名を省略することもできます。省略したときは、CLOAD?が実行されて最初にみつけたプログラム(CSAVEでセーブされたもの)を比較します。

参照

CSAVE、LOAD

CLOSE

機　能

ファイルを閉じます。

書　式

CLOSE [[#]＜ファイル番号＞[,[#]＜ファイル番号＞]･･･]

文　例

```
CLOSE #1
```

解　説

＜ファイル番号＞に対応するファイルを閉じます。指定した＜ファイル番号＞は、以後のOPEN文で指定することができるようになります。また、いったん閉じたファイルをもう一度開くときには、他のファイルが使っていない＜ファイル番号＞ならば、どの＜ファイル番号＞でも指定することができます。

CLOSEでは、＜ファイル番号＞を複数指定することにより、一度に複数のファイルを閉じることができます。また＜ファイル番号＞を省略すると、そのとき開いているファイルをすべて閉じます。

閉じたファイルに対しては、再びOPEN文で開くまで入出力を行うことはできません。

CLOSE文は、ファイルが出力用にオープンされていた場合には、バッファに残っていたデータの掃き出しを行うので、ファイルへの出力処理を正しく終了するためには、必ずCLOSE文を実行しなくてはなりません。

END文、NEW文を実行すると、自動的にすべてのファイルが閉じられますが、STOP文はファイルを閉じないので注意する必要があります。

参　照

OPEN、END、NEW

CLS

機　能

画面に表示されている文字や絵を消します。

書　式

CLS

文　例

```
CLS
```

解　説

CLS は、KEY ON 指定で表示されているファンクションキーの内容を除いて、文字、点、線、絵など画面に表示されているものをすべて消します。
また、CLS が実行されると、テキスト画面の場合、カーソルは画面一番上の左端に移動しますがグラフィック画面では LP（最終参照点）は変化しません。
CLS は漢字モードでは使えません（Illegal function call になる）。漢字モードで画面を消すときは CALL CLS 命令を使います。

参　照

COLOR、CALL CLS

COLOR

機　能

画面の色を指定します。書式1以外は全て MSX2 以降専用です。

書　式 1

COLOR[<前景色>][,<背景色>][,<周辺色>]

書　式 2

COLOR=(<パレット番号>,<R コード>,<G コード>,<B コード>)

書　式 3

COLOR=RESTORE

書　式 4

COLOR[=NEW]

文　例 1

```
COLOR 1,5,8
```

解説 1

色は、それぞれ以下の0〜15（SCREEN 6では0〜3）までの値で指定します。

0 透明（周辺色と同じ色）	8 赤
1 黒	9 明るい赤
2 緑	10 黄
3 明るい緑	11 明るい黄
4 暗い青	12 暗い緑
5 明るい青	13 紫
6 暗い赤	14 灰
7 水色	15 白

電源投入時には、COLORは「COLOR 15,4,7」に設定され、SCREEN文を実行したときには、カラーパレットは上記のように設定されています。
SCREEN 8以外では、書式2のCOLOR =を使ってこれらの色を512色の中から選んで自由に設定することができます。
なお、プログラム中でグラフィック画面を指定すると、それまでテキストモードで使われていた色がそのまま使われます。
SCREEN 8の色は、0〜255が使えます。

0 0 0 0 0 0 0 0	8ビットの色コード
GGGRRRBB	Green、Red、Blueの各ビット

例えば、明るい緑は&B11100000（=224）、暗い紫なら&B00000101（=5）などとなります。
SCREEN 10では0〜15、SCREEN 11、12では0〜255が使えます。SCREEN 10ではYJK/RGB混在モードのRGB部分のみの指定なので、SCREEN 7以下のカラーパレットの指定と全く同じと考えることができます。SCREEN 11以降ではカラーコードはVRAMの値と考えることができますが、YJK方式の性格上これを単純に解釈することはできません。詳しくはV9958のYJKモードに関する記述を参照してください。

■前景色

テキスト画面（SCREEN 0および1のテキストモード）の文字の色や、グラフィック画面の点や線の色です。ただし、グラフィック画面で点や線などを描くときには、各命令で色の指定を行うことができます。各命令で色を指定すると、その指定の方が優先されます。前景色だけを変えるには、例えば、次のように指定します。

```
COLOR 9
```

■背景色
テキスト画面やグラフィック画面の地の色のことです。テキスト画面では、命令実行直後に指定した背景色に変えられます。グラフィック画面では、SCREEN か CLS を実行した後に背景色が変えられます。背景色だけを変えるには、例えば、次のように指定します。

COLOR ,4

■周辺色
画面上下の BASIC で使えない部分のことです。周辺色だけを変えるには、例えば、次のように指定します。

COLOR ,,13

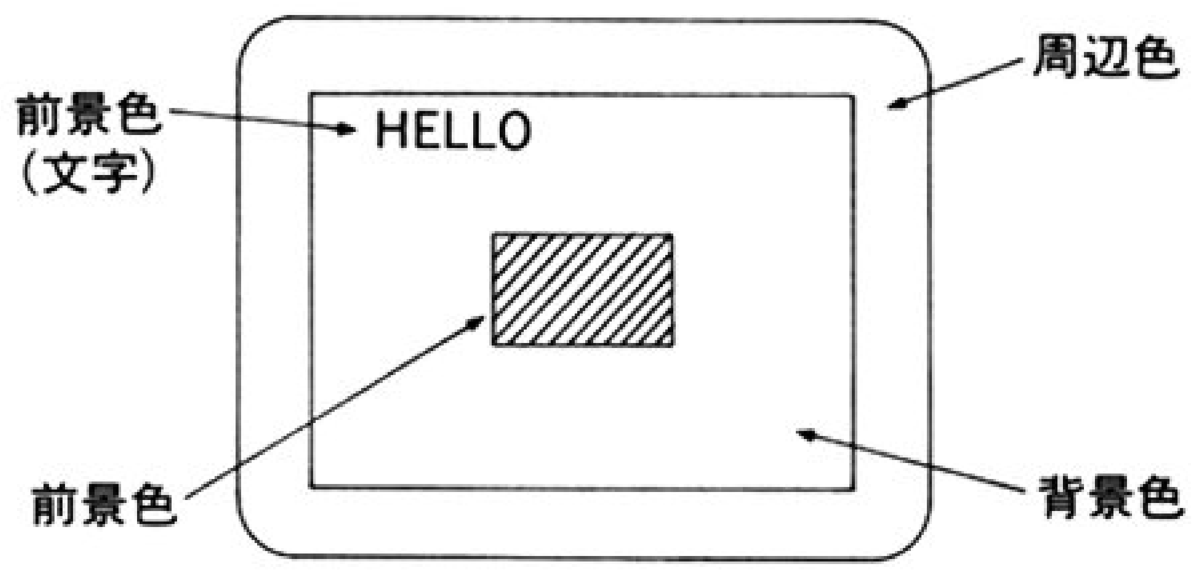


図 2.10 画面色

文 例

```
10 SCREEN 5
20 COLOR = (0,0,7,7)
```

解 説

SCREEN 6 では 0〜3、SCREEN 8 以外では 0〜15 のパレットに色を割り当てます。上の例ではパレット 0 に 0、7、7、つまり

&B000 &B111 &B111
RRR GGG BBB (RGB 各 3 ビット、0〜7 の値)

の結果として明るい水色を割り当てています。割り当てた後は、書式 1 にしたがって色を指定します。
初期状態では、書式 1 で説明したカラー番号とパレット番号が同じになるように設定してあります。

文　例

COLOR = RESTORE

解　説

VRAM 上のパレットテーブルの情報にしたがってパレットを設定します。

文　例

COLOR = NEW

解　説

パレットを初期状態にセットします。= NEW は省略してもかまいません。

参　照

CALL KANJI、CLS、SCREEN、VRAM マップ、V9958

COLOR SPRITE　2

機　能

スプライトの 1 ライン毎の色を決めます。

書 式 1

COLOR SPRITE$(<プレーン番号>)=<文字式>

文 例 1

COLOR SPRITE$(0)=CHR$(1)+CHR$(7)

解　説

スプライトの色も実際には、パレット番号で指定します。この例ではプレーン 0 の 1 ライン目を黒に、2 ライン目を水色にしています。スプライトの 1 ライン毎に色を付けられるのは、SCREEN モード 4 以降のときだけです。色コードについては COLOR を参照してください。なお色コードが 15 を越える場合、上位 4 ビットは次のような意味を持ちます。

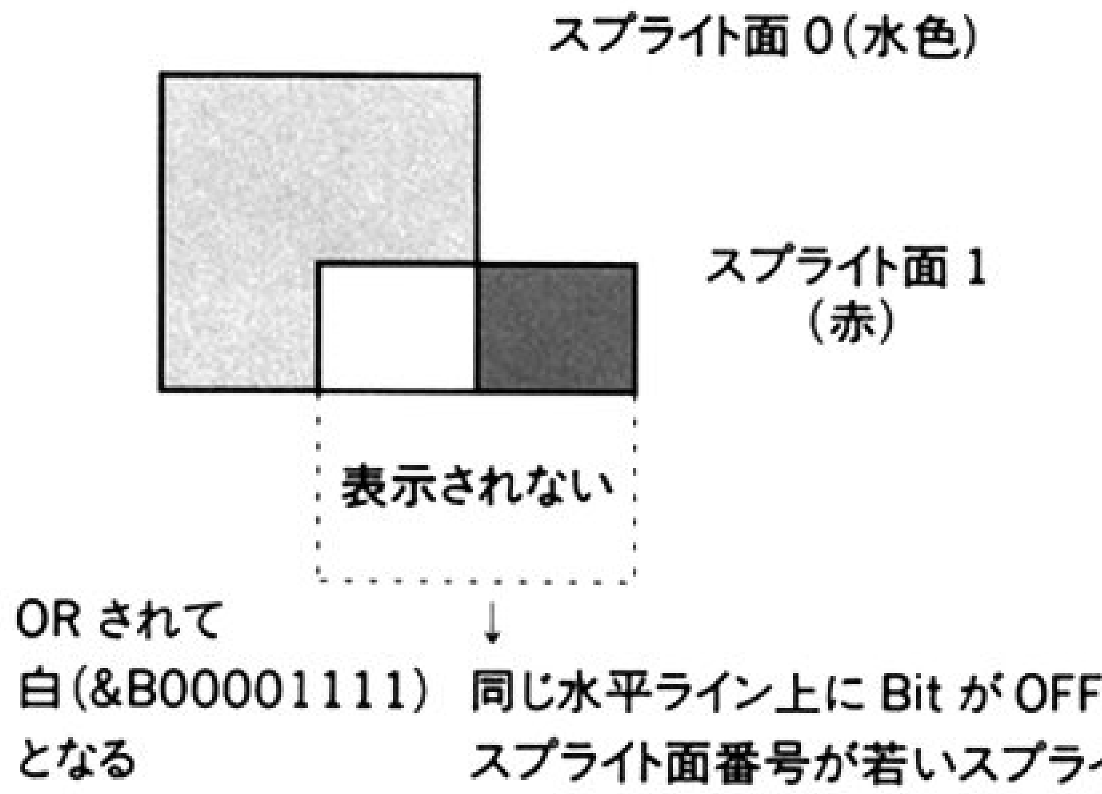
bit7　32 ドット左シフト表示。

bit6　PUT SPRITE を実行したとき連続したプレーンを同時に動かす。このとき重なった部分の色コードは OR されて表示され、衝突は検出されない*。

bit5　　衝突を検出しない。

bit4　　未使用。

*bit6 が ON のときは表示が複雑になります。詳しくは「第 4 部 VDP」を参照して下さい。



すべての SPRITE$を STRING$(32,255)とし、スプライト面 0 が&B00000111（水色)、スプライト面 1 が&B01001000（bit6 が ON された赤）のとき。

図 2.11　bit6 が ON の時のスプライト

bit6 が ON のスプライト面が連続していると、1 つの PUT SPRITE 命令でそれらを動かすことができます。

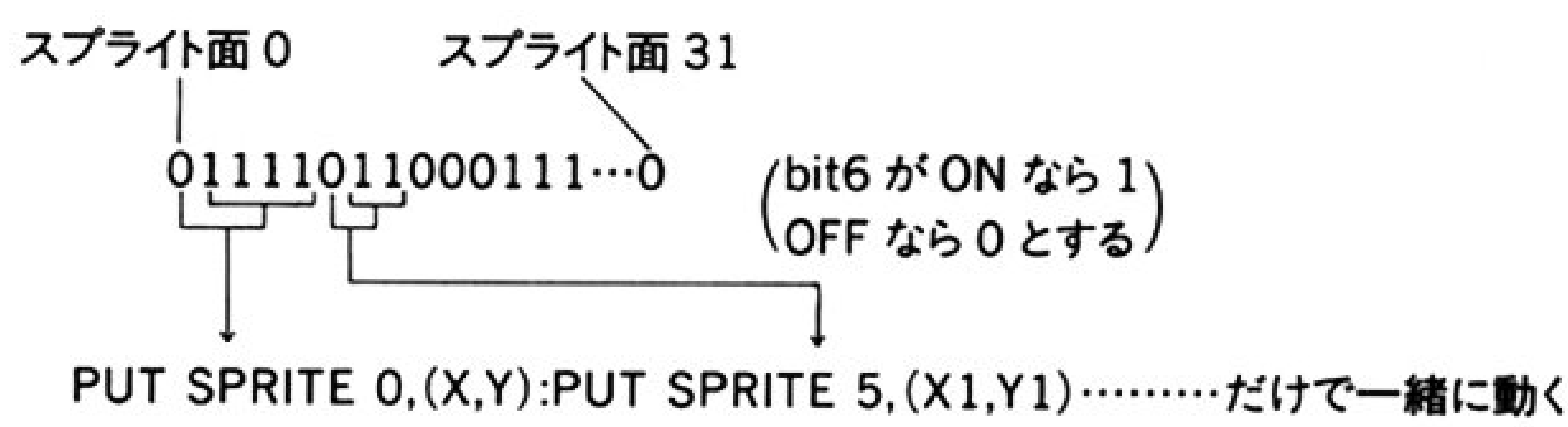


図 2.12　bit6 が ON の時の PUT SPRITE 命令

SCREEN 8 のスプライトだけは、次のような 16 色に固定されます。

表 2.7　SCREEN 8 のスプライト色

カラー番号	0	1	2	3	4	5	6	7
色	黒	暗い青	暗い赤	暗い紫	暗い緑	暗い水色	暗い黄色	灰色
カラー番号	8	9	10	11	12	13	14	15
色	肌色	青	赤	紫	緑	水色	黄	白

書 式 2

COLOR SPRITE(<プレーン番号>)=<式>

文 例 2

```
COLOR SPRITE(1)=15
```

プレーン1全体を白にします。

参 照

COLOR、SPRITE

CONT

機 能

停止したプログラムの実行を再開します。

書 式

CONT

文 例

```
CONT
```

解 説

プログラムの実行停止後、「CONT」と入力すると、停止した次の文から実行が再開されます。プログラムの実行停止は、CTRL + STOP キー入力やプログラム文中のSTOP文やEND文によるものでなければなりません。またINPUT$関数のキー入力待ちをCTRL + STOP キーで止めた場合は、CONTはできません。

実行停止中にLIST命令やPRINT命令などを実行することはできますが、プログラムそのものの変更を行ったときは、CONTによる実行の再開はできません。

参 照

STOP、END

COPY

2 Disk

機 能

ファイルやVRAMの内容、配列をコピーします。書式1以外は全てSCREENモードが5以降で有効なのでMSX2以降専用です。パラメータに<ファイルスペック>を含む書

式は全てDisk BASICでのみ有効です。論理演算子としては、AND、OR、XOR、PSET、PRESETが指定可能です。

書式 1

COPY <ファイルスペック1> [TO <ファイルスペック2>]

書式 2

COPY (<ソース座標1>)−(<ソース座標2>)[,<ソースページ>] TO (<デスティネーション座標>)[,<デスティネーションページ>][,<論理演算子>]

書式 3

COPY (<ソース座標1>)−(<ソース座標2>)[,<ソースページ>] TO <ファイルスペックまたは配列名>

書式 4

COPY <ファイルスペックまたは配列名>[,<方向>] TO (<デスティネーション座標>)[,<デスティネーションページ>][,<論理演算子>]

書式 5

COPY <配列名> TO <ファイルスペック>

書式 6

COPY<ファイルスペック> TO <配列名>

書式 7

COPY SCREEN [<モード>]

文例 1

```
COPY "A:SAMPLE.DAT" TO "SAMPLE.BAK"
```

解説 1

<ファイルスペック1>を<ファイルスペック2>で指定したドライブへ、指定したファイル名でコピーします。このCOPYコマンドはディスク装置間のみで有効です。
指定の方法は4種類あります。

- **<ファイルスペック1>だけを指定した場合**

  COPY "B:SAMPLE.DAT"

  この場合、SAMPLE.DATは、カレントドライブに同じ名前(SAMPLE.DAT)でコピーされます。

■ <ファイルスペック2>でドライブ名だけを指定した場合
COPY "A:SAMPLE.DAT" TO "B:"
この場合、SAMPLE.DATは、ドライブBに同じ名前でコピーされます。

■ <ファイルスペック2>でファイル名だけを指定した場合
COPY "B:SAMPLE.DAT" TO "SAMPLE.BAK"
この場合、SAMPLE.DATは、カレントドライブにSAMPLE.BAKという名前でコピーされます。

■ <ファイルスペック2>でドライブ名とファイル名とを指定した場合
COPY "A:SAMPLE.DAT" TO "B:SAMPLE.BAK"
この場合、SAMPLE.DATは、ドライブBにSAMPLE.BAKという名前でコピーされます。

文例2

```
10 SCREEN 5
20 COPY(10,10)-(100,120),1 TO (30,30),0,XOR
```

解説2

VRAM上の長方形の領域を他の領域にコピーします。ページを省略した場合はそれぞれアクティブページとみなされます。論理演算子を省略した場合はPSETとみなされます。
COPY命令で画面範囲外にコピーした場合、スプライトが表示されてしまうことがあります。例えば、

```
COPY(0,0)-(255,39) TO (0,211)
```

を実行すると画面データをスプライトなどのデータエリアに書いてしまうので、スプライトのゴミが表示されてしまいます。VDP(V9938、V9958)はハードウェアスクロールの機能を持っており、COPY文では、その機能を十分生かすために画面表示範囲外に画面データをコピーできるようになっています。したがって、画面表示範囲外に画面データをコピーするときには、スプライトをVDP命令で消して下さい。VDPレジスタのR#8のbit 1を1にすると、スプライトの表示が禁止されます。具体的には、以下の命令を実行します。

```
VDP(9)=VDP(9) OR 2
```

文例3

```
10 SCREEN 8
20 DIM B%(327)
30 COPY(100,100)-(120,130),0 TO B%
```

解説3

VRAM上の長方形の領域の内容を配列やファイルにコピーします。配列の場合は必要な大きさ以上のものを確保しておかなければなりません。必要なバイト数Nは次のように計算されます。

N = INT(X*Y/K+5)

ただしXは横、Yは縦のドット数、Kは定数で、

SCREEN 5と7では2
SCREEN 6では4
SCREEN 8以降では1

として求め、次に配列の型によって

整数型	(N+1)¥2
単精度型	(N+3)¥4
倍精度型	(N+7)¥8

として最低限必要な要素数が得られます。文例3のケースでは

N = INT((120-100+1) * (130-100+1) / 1+5):N = 656
(N+1)¥2 = 328

からDIM B%(327)となります。

文例4

```
10 SCREEN 7
20 COPY "TEST",1 TO (100,100),1,AND
```

解説4

ファイルまたは配列の内容を画面に転送します。方向はデータを書き込む向きで、

0	左上から右下
1	右上から左下
2	左下から右上
3	右下から左上

のどれかです。

文例 5

```
10 DIM A%(100)
20 COPY A% TO "TEST"
```

解説 5

配列の内容をファイルにセーブします。

文例 6

```
10 DIM A%(100)
20 COPY "TEST" TO A%
```

解説 6

ファイルから配列に内容をコピーしてきます。ファイルはもともと書式3や書式5で作ったものでなければならず、配列は十分な大きさでなければなりません。

文例 7

```
10 SCREEN 8
20 COPY SCREEN
```

解説 7

外部ビデオ信号をデジタル化し、VRAMに送ります。＜モード＞は0なら通常で、1ならインタレースモードです。この機能は画像取り込み装置がつながっている場合にのみ有効です。どのスクリーンでどのモードで取り込みができるかはその取り込み装置の性能に依存します。

参照

SET PAGE、LINE

COS

機能

＜数式＞の余弦（コサイン）を計算します。

書式

COS(＜数式＞)

文 例

```
PRINT COS(3.14159)
```

表示 −.99999999999649

CSAVE

機 能

メモリ上にあるプログラムをカセットテープにセーブします。

書 式

CSAVE<ファイル名>[,<ボーレート>]

文 例

```
CSAVE "TEST"
```

解 説

メモリ上にあるプログラム(LIST命令で画面に表示されるもの)を、カセットテープに保存します。
ファイル名の後に次のように指定すれば、セーブするときの速さを指定することができます。指定できるのは1か2です。

```
CSAVE "TEST",1
```

2と指定すると1の倍の速さでセーブされますが、カセットレコーダの性能によってはうまくセーブできないことがあります。このセーブの速さは、SCREEN命令でも指定でき、CSAVEで省略されたときには、SCREEN命令の指定が使われます。
うまくセーブされたかどうかは、CLOAD?命令で確認してください。セーブされたプログラムは、CLOAD命令を使ってロードすることができます。
CSAVEとSAVEとの違いについては、SAVE命令を参照してください。

参 照

CLOAD、CLOAD?、SAVE

CSNG

機　能

＜数式＞の値を単精度実数に変換します。

書　式

CSNG(＜数式＞)

文　例

```
PRINT CSNG(123.45678)
```

表示 123.456

CSRLIN

機　能

テキスト画面上のカーソルの行位置を返します。

書　式

CSRLIN

文　例

```
A = CSRLIN
```

解　説

CSRLIN は、現在カーソルが何行目にあるのかを、一番上の行を 0 とした値で調べます。
CSRLIN は、テキスト画面でのみ使うことができます。

参　照

POS

CVI、CVS、CVD

Disk

機　能

文字列を数値データに変換した値を返します。

書　式

CVI(<2バイト文字列>)

CVS(<4バイト文字列>)

CVD(<8バイト文字列>)

文　例

```
PRINT CVI(A$)
```

解　説

ディスク上のランダムファイルから読み込んだ数値データは文字型になっているため、これらの関数を使って数値型に変換しなければなりません。CVIは2文字(2バイト=16ビット)の文字列を整数値に、CVSは4文字の文字列を単精度実数値に、CVDは8文字の文字列を倍精度実数値にそれぞれ変換します。

この変換は、例えば、CVIの場合、次のように行われます。

A$="90"の場合、すなわちA$=CHR$(&H39)+CHR$(&H30)の場合には、CVI(A$)の値は、キャラクタコードを数値データとみなし、&H3039 (=12345) となります。

CVS、CVDの場合は、各キャラクタコードを正規化された浮動小数点形式の数値データとみなします。

DATA

機　能

READ文で読み込むためのデータをプログラム中に用意します。

書　式

DATA<定数>[,<定数>・・・]

文　例

```
10 READ A,B
20 DATA 1,2
```

解 説

<定数>は文字型か数値型で、READ文でそのデータを読み込む変数の型と一致していなければなりません。この例では数値型の変数AおよびBに、それぞれ1と2という数値が代入されます。

DATA文中のデータは、1行に収まる範囲内でいくつでもカンマで区切って記述することができます。DATA文中のコロン（：）は引用符に囲まれていない限りマルチステートメントの区切りと解釈されます。DATA文は、READ文によって読み込まれない限り何の働きもしません。

参 照

READ、RESTORE

DEF FN

機 能

計算式を定義します。

書 式

<名前>[(<引数>[,<引数>・・・])]=<関数の定義式>

文 例

```
10 DEF FN A(X)=X*X*3.14
20 FOR I=10 TO 20 STEP 5
30 PRINT FN A(I)
40 NEXT
```

表示
```
314
706.5
1256
```

解 説

10行で円の面積を出す計算式をFN A(X)として定義し、その式を30行でIという変数を使って値を変えながら呼び出しています。

DEF FNは、名前（上の場合A）で、計算式を定義しておくことができます。名前は、英字で始まる2文字以内の英数字で付けます。名前の次にカッコで囲んだ変数（上の場合X）を指定して、計算式（上の場合X*X*3.14）を定義します。カッコ内の変数と計算式の中の変数は対応しています。

定義した計算式は「FN 名前（変数)」という形で呼び出せます。定義を行ったときの変数は仮のものなので、呼び出すときにはカッコ内でそのときどきで必要な変数（上の場合I）に変えて指定することができます。

DEFINT / SNG / DBL / STR

機　能

変数の型を宣言します。

書　式

DEFINT ＜文字の範囲＞[, ＜文字の範囲＞・・・]
DEFSNG ＜文字の範囲＞[, ＜文字の範囲＞・・・]
DEFDBL ＜文字の範囲＞[, ＜文字の範囲＞・・・]
DEFSTR ＜文字の範囲＞[, ＜文字の範囲＞・・・]

文　例

```
10 DEFINT A,B
20 DEFSTR C-E
30 A = 3.12:B = 0.33
40 C =''3.12'':D =''0.33'':E =''ABC''
50 PRINT A,B
60 PRINT C,D,E
```

表示

```
3         0
3.12      0.33      ABC
```

解　説

この例では変数A、Bを整数型に、変数C、D、Eを文字型に宣言しています。変数の指定は、英字1字（例 A）あるいは英字1字-英字1字（例 A-Z）という形で行います。型は以下のように設定されます。

DEFINT	整数型
DEFSNG	単精度実数型
DEFDBL	倍精度実数型
DEFSTR	文字型

型宣言が行われていない変数は、すべて倍精度実数とみなされます。また、DEF 命令での型宣言よりも、型宣言文字での指定の方が優先されます。

参　照

CLEAR

DEF USR

機　能

機械語プログラムの実行開始番地を指定します。

書　式

DEF USR[<番号>]=<開始番地>

文　例

```
DEF USR2 =&HE000
```

解　説

USR を使って呼び出す機械語プログラムの、実行開始番地（&HE000）を指定します。DEF USR の次には、複数の機械語プログラムを使うときに区別するための番号（2）を付けます。指定できるのは、0 から 9 までの値です。

参　照

CLEAR、USR、「5 章　マシン語とのリンク」

DELETE

機　能

指定されたプログラム行を削除します。

書　式

DELETE<行番号の範囲>

文　例

```
DELETE 50-100
```

解 説

行番号の指定の方法によって、さまざまな形の削除が行えます。以下に、その例を示します。

DELETE 10

行番号 10 の行を削除します。

DELETE -100

プログラムの先頭から行番号 100 までの行を削除します。

参 照

LIST、MERGE、NEW、RENUM

DIM

機 能

配列変数を使うことを宣言し、それをメモリ領域に割り当てます。

書 式

DIM＜変数名＞(＜添字の最大値＞[,＜添字の最大値＞・・・])

文 例

```
DIM A(11),B%(100),C$(2,3)
```

解 説

DIM は、使用する配列変数の最大値を設定し、同時にメモリ上にその配列の領域を確保します。設定できる最小の値は 0、最大の値はメモリの容量が許す範囲です。設定を行わない場合は、値は 10（要素数は添え字 0〜10 で 11 個）とみなされます。
不用になった配列変数は、ERASE 命令で削除できます。また、宣言してある配列変数は、CLEAR 命令か ERASE 命令を実行しないと再宣言できません。

参 照

ERASE、CLEAR

DRAW

機 能

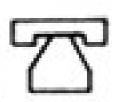
グラフィック画面に図形を描きます。

書 式

DRAW＜文字式＞

文 例

```
10 SCREEN 2
20 A$="A0C15S4BM150,80L40D12R8U6R24D6R8U12"
30 B$="BM134,86F12D8L32U8E12"
40 C$="C8BM132,92L4G4D4F4R4E4U4H4"
50 DRAW "XA$;":DRAW "XB$;"
60 DRAW XC$
70 GOTO 70
```

表示 

解 説

各文字変数に代入された文字式が、50行の指定で実行されて、画面に電話器が描かれます。

DRAWでの指定は次のような形で行います。

DRAW "文字式"
DRAW 文字変数
DRAW "X文字変数；"

文字式は、線を描くための情報で記号と数値でできていて、DRAWに続けて引用符で囲んで指定します。文字変数を使って指定するときは、あらかじめその文字変数に、線を描くための情報（文字式）を代入しておきます。文字変数を用いるときには、"X文字変数；"と指定しても（50行）、単に文字変数だけ指定しても（60行）、結果は同じです。

指定する記号には、移動、回転、色、サイズの4種類のものがあります。

表 2.8 DRAW 文の指定一覧

	記号	数値	意 味	例	備 考
移動命令	U	距離	上へ移動	U10	距離は画面の 1 点を単位とする(単位はサイズ命令で変更可能)斜め方向と上下左右方向で同じ距離を指定しても、画面上での実際の移動距離は異なる。
	D	〃	下 〃	D5	
	L	〃	左 〃	L8	
	R	〃	右 〃	R12	
	E	〃	右上 〃	E7	
	F	〃	右下 〃	F20	
	G	〃	左下 〃	G15	
	H	〃	左上 〃	H40	
	M	座 標 (X,Y)	(X,Y) まで描画	M100,100	X の前に＋か－をつけると相対座標指定
	B	なし	移動のみ	BM50,50	U～M までの記号に付けて用いる。
	N	なし	LP 移動しない	NL10	U～M までの記号に付けて用いる。
回転命令	A	0～3	90 度単位で回転	A1U12L4D12	0～270 度の回転。指定し直すまで有効。
色命令	C	0～255	色指定	C8D10	色コードの範囲は画面モードによる。
サイズ命令	S	1～255	移動距離の単位	S20U12L4D12	省略値は 4 で、これが画面 1 ドット分。

■ 数値変数の指定

なお、次の形式で数値変数も指定できます。

DRAW "=変数名；"

例えば、

```
10 SCREEN 2
20 A = 100:B = 120
30 DRAW "BM = A;,= B;U20"
40 GOTO 40
```

というプログラムは (100,120) の位置まで線を引かずに移動し、そこから上に 20 点分の線を引くというものです。

参　照

COLOR

DSKF

Disk

機　能

フロッピーディスクの残り容量をクラスタ単位で返します。

書　式

DSKF（<ドライブ番号>）

文　例

```
PRINT DSKF(1)
```

解　説

<ドライブ番号>で指定されたフロッピーディスクの残り容量をクラスタ単位で返します。1 クラスタは以下のとおりです。

メディア	サイズ
フロッピーディスク	1024 バイト
RAM ディスク*	512 バイト（2M バイトまで） 1024 バイト（2M～4M バイト）
ハードディスク	可変〔ハードディスクの初期化（領域確保）のとき、指定する領域の大きさにより 1K バイト、2K バイト、4K バイト、8K バイト、16K バイト、32K バイト、64K バイトから選択する。〕

* MSX-DOS2 の RAMDISK コマンドで設定するもの。

END

機　能

プログラムを終了します。

書　式

END

文　例

```
10 PRINT "ハルカ゛ "
20 END
30 PRINT "コナイ"
40 END
```

解　説

END命令はプログラムを終了させます。プログラム内にいくつ置いてもかまいません。また、プログラムの最後のEND（この場合、40行）は省略してもかまいません。
上の例では、20行にENDがあるので、そのあとの行は実行されず、「コナイ」は表示されません。

EOF

機　能

ファイルが終了したかどうかを調べます。

書　式

EOF（<ファイル番号>）

文　例

```
IF EOF(1) THEN CLOSE1
```

解　説

EOFは、ファイル番号で指定されたファイルが終ったかどうか調べるもので、終っていれば−1を、まだ途中であれば0を結果として返します。EOFに続くカッコ内の数値は、指定したファイルのファイル番号で、OPEN命令で指定されているものと対応しています。調べるファイルは、OPEN命令のINPUTモードで開かれていなければなりません。

参　照

OPEN、INPUT #

ERASE

機能

配列変数を削除します。

書式

ERASE<配列変数名>[,<配列変数名>・・・]

文例

```
ERASE A
```

解説

DIMで指定した配列変数を削除し、割り当てられていたメモリを使えるようにします。

参照

DIM

ERL、ERR

機能

エラーコードやエラーの起こった行番号を調べます。

書式

ERR

ERL

文例

```
10 ON ERROR GOTO 40
20 A%=A%+10
30 GOTO 20
40 IF ERR<>6 THEN 80
50 PRINT ERL;"ｷﾞｮｳﾃﾞ ｴﾗｰﾊｯｾｲ"
60 A%=0:PRINT "ｼｮﾘｼﾏｼﾀ"
70 RESUME
80 ON ERROR GOTO 0
```

解　説

40行で発生したエラーコードが6かどうかの判定を行い、6なら50行でエラーの発生した行番号を表示し、60行でエラーに対する処理を行っています。
このように、ERLやERRを用いると、エラーが発生したとき、エラーの起こった行番号（ERL）やエラーコード（ERR）を知ることができます。
ERLやERRは普通、ON ERROR GOTO命令でジャンプするエラー処理の操作で使われます。ダイレクトモードでエラーが発生したときには、ERLの値は65535となります。

参　照

ON ERROR GOTO、ERROR

ERROR

機　能

プログラムをエラー状態にします。

書　式

ERROR＜エラーコード＞

文　例

```
ERROR 23
```

解　説

ERRORで0から255までの値を指定することによって、プログラムをエラー状態にすることができます。メッセージが用意されているエラーコードを指定すると、画面に対応するエラーメッセージが表示されます。
エラーコード23、26～49、60～255にはメッセージが定義されていないので、ON ERROR GOTO文と併せて独自のエラー処理を行うこともできます。

参　照

ON ERROR GOTO、エラーメッセージ表

EXP

機　能

自然対数の底「e」の＜数式＞乗を計算します。

書　式

EXP(＜数式＞)

文　例

```
PRINT EXP(1)
```

表示 2.7182818284588

FIELD

Disk

機　能

ランダム入出力バッファ中に変数領域を割り当てます。

書　式

FIELD[#]＜ファイル番号＞,＜フィールド幅＞AS＜文字変数＞[,＜フィールド幅＞AS＜文字変数＞]・・・

文　例

```
FIELD # 1, 15 AS A$, 10 AS B$
```

解　説

＜ファイル番号＞とはOPEN文で開いたファイルにつけた番号のことで＜フィールド幅＞とは＜文字変数＞に割り当てられる文字数のことです。また＜文字変数＞はランダムファイルをアクセスするとき使われる変数名です。

FIELD文は、GET文によるランダム入出力用バッファへのデータの読み込み、あるいはPUT文によるランダム入出力用バッファからのデータの書き出しを可能にするためにGET、PUT文の前に実行されなければなりません。

例えば、

```
FIELD1, 15 AS A$, 10 AS B$
```

では、ランダム入出力用バッファ中の最初の15文字（バイト）をA$という文字変数に

割り当て、次の10文字がB$に割り当てられます。
FIELD文は、実際にはランダム入出力用バッファにデータを置くことはせず、これはLSET、RSET文で行われます。
FIELD文で割り当てられる総バイト数は、OPEN時に決めたレコード長を越えることはできません。万一、越えた場合には、「Field overflow」エラーが生じます。
なお、1つのファイル番号に対して、何回かに分けてFIELD文を実行してもかまいません。このとき、以前に実行されたFIELD文も、その効力を保持します。FIELD文は、実行するたびに最初の1文字からバッファを定義するため、同一データに対し複数の領域を定義する効果をもちます。
FIELD文で定義した文字変数は、ランダム入出力用バッファ中の対応するデータを指していますが、FIELD文の実行後に入力関係の文、または代入文の左辺で使用すると、その変数は文字列の記憶領域に移り、ランダム入出力用バッファからは消えてしまうので注意する必要があります。

FILES、LFILES　Disk

機　能

フロッピーディスクの中のファイル名を表示します。

書　式

[L]FILES [<ファイルスペック>]

文　例

```
FILES "*.BAS"
LFILES "B:"
```

解　説

<ファイルスペック>で指定されたファイルのファイル名を表示します。指定ファイルがフロッピーディスク中になければ、「File not found」エラーが生じます。
<ファイルスペック>が省略された場合は、現在選択されているディスクドライブ中の全ファイル名を表示します。
指定するファイル名には、疑問符(?)を含むことができ、ファイル名や、拡張子中の1文字の代わりをすることができます。また、ファイル名、拡張子の文字中にアスタリスク(*)を用いると、任意のファイル名、または拡張子の代わりとすることができます。
<ファイルスペック>で、ディスクドライブが指定されていれば、そのディスクドライブ中の指定ファイルが出力され、そうでないときは、現在選択されているディスクドラ

イブ（カレントドライブ）が使用されます。

LFILES は出力先がプリンタであることを除き、FILES とまったく同じです。

FIX

機　能

<数式>の値の小数点以下を取り去った値を計算します。

書　式

FIX(<数式>)

文　例

```
PRINT FIX(−1.2)
```

表示 −1

解　説

<数式>の値が正のときは INT と同じですが、負のときは以下のようになります。

式	結果	意　味
FIX（−1.2）	−1	小数点以下切り捨て
INT（−1.2）	−2	−2.3 を超えてない最大の整数

FOR～NEXT

機　能

FOR から NEXT までの間で指定された操作を、指定された回数だけ繰り返します。

書　式

FOR<変数名>=<初期値>TO<終値>[STEP<増分>]

文　例

```
10 FOR I=0 TO 10 STEP 2
20 PRINT I
```

```
30 PRINT "ループ "
40 NEXT I
50 END
```

解 説

FORで繰り返しの回数を指定し、FORとNEXTの間の操作をその回数だけ繰り返し行います。FORの次に「I＝0」の形で、繰り返しの初期値（0）を設定します。TOの次で繰り返しの最終値（10）を設定します。最後にSTEPに続けて初期値と最終値との間の増分（2）を指定します。

プログラムの実行がNEXT（40行）まで来ると、変数Iが増分だけ増やされます。そして、最終値を超えていなければFOR（10行）に戻ります。したがって、初期値が最終値よりも大きく設定されているときも、FOR文からNEXT文までのプログラムは1度実行されます。

このプログラムでは、10行の指定で、Iが0、2、4、6、8、10と増えますので、20行と30行は6回実行され、「ループ」は変数Iの値とともに6回表示されます。Iが最終値10と同じになったところでプログラムの実行はFOR～NEXTから出て、NEXTの次の行に移ります。

FOR～NEXTのSTEP以下は省略することができます。省略したときは増分は＋1になります。

FRE

機 能

メモリの未使用領域の大きさを返します。

書 式

FRE(<引数>)

文 例 1

```
PRINT FRE(0)
```

解 説 1

引数が数値の場合は、BASICが使っていないメモリの大きさ(ユーザーエリアの未使用領域）がバイト数で返されます。引数は、数値であれば何でもかまいません。

文 例 2

```
PRINT FRE("A")
```

解 説 2

引数が文字列の場合は、まず文字領域の整理が行われ、それから未使用の文字領域の大きさがバイト数で出されます。この引数は、文字列であれば何でも（空文字でも）かまいません。

参 照

CLEAR、メモリマップ

GET

Disk

機 能

ランダムファイルから、ランダム入出力用バッファに1レコードを読み込みます。

書 式

GET［#］<ファイル番号>[,レコード番号>]

文 例

```
GET #1,1
```

解 説

<ファイル番号>は、そのファイルをOPEN文によって開いたときに指定した番号です。<レコード番号>が省略された場合には、直前のGET、PUT文で参照されたレコードの次のレコードがバッファに読み込まれます。レコード番号として指定可能な最大の数は、4,294,967,295です。

GET DATE

2

機 能

<文字変数>に現在の日付を代入します。

書　式

GET DATE ＜文字変数＞[,A]

文　例

GET DATE A$

解　説

現在の日付を指定した＜文字変数＞に代入します。オプションのAを付けると、アラームの設定された日付が代入されます。日付は、「85/04/03」のように年月日がスラッシュで区切られ、月日が1桁の場合は04のように2桁となります。

GET TIME　2

機　能

＜文字変数＞に現在の時刻を代入します。

書　式

GET TIME　＜文字変数＞[,A]

文　例

GET TIME T$

解　説

現在の時刻を指定した＜文字変数＞に代入します。オプションのAを付けると、アラームの設定された時刻が代入されます。時刻は、「11:20:03」のように時分秒がコロン(：)で区切られ、時刻が1桁の場合は03のように2桁となります。

GOSUB

機　能

指定したサブルーチンを呼び出します。

書　式

GOSUB＜行番号＞

文　例

```
10 GOSUB 30
20 END
30 PRINT "ココカラサフ゛ルーチン"
40 PLAY "CDE"
50 PRINT "GOSUB ノツキ゛ノキ゛ョウヘモト゛ル
60 RETURN
```

解　説

GOSUB命令で指定した行で始まるサブルーチンを呼び出します。呼び出されたサブルーチンは、RETURN命令によってGOSUBの次の文または行に戻ります。サブルーチンとは、GOSUBで指定する行から、RETURNの行までの独立した1つのプログラムです。

参　照

RETURN、GOTO

GOTO

機　能

指定した行にジャンプします。

書　式

GOTO<行番号>

文　例

```
10 PRINT "10 キ゛ョウテ゛ス":GOTO 30
20 PRINT "20 キ゛ョウテ゛ス":END
30 PRINT "30 キ゛ョウテ゛ス"
40 GOTO 20
```

表示
```
10 キ゛ョウテ゛ス
30 キ゛ョウテ゛ス
20 キ゛ョウテ゛ス
```

解　説

10 行で、30 行に飛ぶように指定しているので、20 行の END 命令を飛び越して 30 行が実行されます。そのあと、40 行から 20 行にジャンプして、プログラムは終了します。

参　照

ON GOTO、ON GOSUB、GOSUB

HEX$

機　能

<数式>の数値を 16 進数の文字列に変換します。

書　式

HEX$(<数式>)

文　例

```
PRINT HEX$(12)
```

表示 C

参　照

VAL、BIN$、OCT$、STR$

IF～THEN～ELSE、IF～GOTO～ELSE

機　能

条件判断を行い、その結果に応じた操作を行います。

書　式

IF<条件>THEN<文または行番号>[ELSE<文または行番号>]

IF<条件>GOTO<行番号>[ELSE<文または行番号>]

文　例

```
10 A=A+1
20 PRINT A
```

```
30 IF A = 20 THEN PRINT "ハタチ" ELSE 10
40 END
```

解　説

IF に続けて条件を指定します。その条件が満たされた（0 以外の値を取る）ときは THEN 以下の指定が実行され、条件が満たされない（0 を値として取る）ときは ELSE 以下の指定が実行されます。

ELSE 以下の指定は省略することができます。その場合、条件が満たされなかったときは次の行が実行されます。

THEN、ELSE の後ろの指定は、文でも行番号でもかまいません。THEN の後ろに行番号を指定する場合、THEN の代わりに GOTO を使って、IF～GOTO～ELSE という形にすることもできます。

条件には、以下のような比較演算子と論理演算子とが使えます。論理演算子は、複数の条件を指定するときに用います。

比較演算は条件が成立したときには「−1」、成立しないときには「0」を値として返します。

表 2.9　比較演算

比較演算子	意　味	例
式 1 = 式 2	式 1 が式 2 と等しい	IF A$="y"
式 1 <> 式 2	式 1 が式 2 と等しくない	IF A<>0
式 1 < 式 2	式 1 が式 2 より小さい	IF A<10
式 1 > 式 2	式 1 が式 2 より大きい	IF A>B
式 1 <= 式 2 (式 1 =< 式 2)	式 1 が式 2 と等しいか、小さい	IF A<=B+1
式 1 >= 式 2 (式 1 => 式 2)	式 1 が式 2 と等しいか、式 2 より大きい	IF A$>="Z"

表 2.10　論理演算

論理演算子	例	意　味
NOT	IF NOT (A = 1)	A が 1 でなければ THEN 以下を実行
AND	IF A = 0 AND B = 3	A が 0 で B が 3 のときのみ THEN 以下を実行
OR	IF A<0 OR B = 0	A が 0 より小さいか、B が 0 のときに THEN 以下を実行

INKEY$

機　能

キーが押されていればその文字を、押されていなければ空文字（ヌル）を返します。

書　式

INKEY$

文　例

```
10 A$= INKEY$
20 IF A$="" THEN 10
30 PRINT A$;"=";HEX$(ASC(A$))
40 GOTO 10
```

解　説

このプログラムを実行すると、キーボードから打ち込まれた文字と、その文字のアスキーコードが画面に表示されます。
INKEY$は、[CTRL]＋[C]と[CTRL]＋[STOP]以外のコントロール文字も読み取ります。

参　照

INPUT、LINE INPUT、INPUT$、キャラクタコード表

INP

機　能

指定した入力ポートからデータを獲得します。

書　式

INP(<ポート番号>)

文　例

```
A = INP(15)
```

解　説

カッコ内のポート番号 (15) で指定された入力ポートから1バイトのデータを読み取り、その値をAに代入します。ポート番号に関しては、巻末の「I/Oマップ」を参照して下さい。

INP関数は、ハードウエアと非常に密着した関数で、他のMSX機種では返す値が違う場合もあります。MSXの互換性を維持するため、広く公表するプログラムには使用しないでください。

参　照

OUT、WAIT

INPUT

機　能

キーボードから入力されるデータを指定した変数へ代入します。

書　式

INPUT["<プロンプト文>";]<変数名>[,<変数名>・・・]

文　例

```
10 INPUT "データ";A
20 PRINT A * A:GOTO 10
```

解　説

INPUT命令を実行すると、疑問符 (?) が画面に表示され、プログラムはキーボードからのデータの入力待ちになります。何か数値を入力するとその数値は?に続いて画面に表示され、その値が指定された変数Aに代入されます。

引用符 (") で囲まれた文字列 (上の例の場合"データ") は、プロンプト文と呼ばれるもので、これを指定していれば、?の前にその文字列が表示されて、メッセージの役割を果たします。

変数は、カンマ(,)で区切って複数個指定することもできます。この場合、入力するデータもカンマで区切って、変数の数だけ入力しなければなりません。入力するデータの個数が足りないと、「??」と表示されて入力待ちとなり、個数が多いと「?Extra ignored」と表示され、余分に入力されたデータは無視されます。また、対応する変数の型とデータの型は、一致していなければなりません。型が違っていた場合には、「Redo from start」と表示されて再び入力待ちとなります。

INPUT

機 能

ファイルからデータを読み込みます。

書 式

INPUT #<ファイル番号>, <変数>[, <変数>･･･]

文 例

INPUT # 1,A,B

解 説

あらかじめOPEN文でINPUTモードを指定して開いておいたファイルからデータを読み込み、変数にセットします。INPUT #文はデータを読み込む対象がファイルであることを除けば、INPUT文とほぼ同じです。

<ファイル番号>は、OPEN文で指定した番号です。

INPUT #文で読み込むデータは、PRINT #文で書き出した各データと対応した順に読み込まれます。したがって<変数>の型はファイルから入力するデータの型と対応していなくてはなりません。

ファイル中のデータのうち、先に続くスペース、ラインフィードは無視され、スペース、ラインフィード以外の最初の文字がデータの始まりとされます。データは、スペース、キャリッジリターン、ラインフィード、カンマによって区切られていなければなりません。

ファイル中のデータが文字型の場合には、最初の文字が引用符（"）であると、その文字型データは、最初の引用から次の引用符までの間に読み込まれた文字で構成されているとみなされます。したがって、引用符で囲まれた文字列中には、引用符を文字として入れることはできません。最初の文字が引用符以外のときは、文字列は、囲まれてないものとして扱われ、カンマ、キャリッジリターン、ラインフィード、または255文字の読み込み終了によって区切られます。数値、または文字型データの読み込み中に、ファイルの終わり（EOF）に達したときは、そのデータ項目はそこで区切られます。

参 照

INPUT、OPEN、PRINT #

INPUT$

機　能

指定されたファイルから指定された長さの文字を獲得します。

書　式

INPUT$(<文字列>[,[#]<ファイル番号>])

文　例

WD$= INPUT$(6,# 2)

解　説

<ファイル番号>で指定されたファイルから<文字数>分の文字列を読み出します。
INPUT$は、CTRL + STOP、CTRL + C を除くすべての文字をそのまま読み出すので、INPUT 文や LINE INPUT 文では入力することのできないラインフィードコードやリターンコードも入力することができます。
#のあとの数値は、読み出すファイルのファイル番号で、OPEN 命令で指定されたものと対応しています。ファイル番号が省略された場合には、キーボードから入力されたデータを読み出しますが、INPUT 命令と異なり、入力された文字は画面に表示されません。
CTRL + STOP でこの入力を中断した場合、CONT は実行できません。
カッコ内の最初の数値は、読み出す文字数を指定しています。キーボードからの入力を読み出す場合、INPUT$は指定された文字数の文字が入力されるのを待ち続けますが、すでにキーボードバッファに入力済みのデータがあるときには、バッファの中の文字から先に読み出します。

参　照

OPEN、INPUT、LINE INPUT

INSTR

機　能

ある文字列の中から指定された文字列を捜し、見つかった位置を返します。

書　式

INSTR([<数式>,]<文字列 1>, <文字列 2>)

文例

```
10 A$="SUPER MSX"
20 PRINT INSTR(A$,"M")
30 END
```

表示 7

解説

引数の最初の文字列（A$）の中から、2番目の文字列（「M」）をさがします。この例の場合、「M」はA$の頭から数えて7番目にあるので、結果は7になります。
指定された文字列が見つからなかったときは、結果は0になります。また、INSTR (6, A$,"M")のように、最初に数値を入れて、さがし始める位置を指定することもできます。

INT

機能

<数式>以下の最大の整数を返します。

書式

INT(<数式>)

文例

```
PRINT INT(-1.2)
```

表示 −2

解説

<数式>の値が正のときはFIXと同じですが、負のときは異なります。FIXを参照して下さい。

INTERVAL ON / OFF / STOP

機能

ON INTERVAL GOSUB命令の設定を実行するかどうか決定します。

書　式

INTERVAL ON
INTERVAL OFF
INTERVAL STOP

文　例

```
ON INTERVAL = 100 GOSUB 100:INTERVAL ON
```

解　説

INTERVAL ON は、ON INTERVAL GOSUB の設定を実行するように指定し、以後、指定した時間ごとにサブルーチンにジャンプします。INTERVAL OFF は、設定によるジャンプを禁止します。INTERVAL STOP は、設定によるジャンプを一時止め、後で INTERVAL ON が指定されると直ちにジャンプを実行します。

参　照

ON INTERVAL GOSUB

KEY

機　能

ファンクションキーの内容を定義します。

書　式

KEY<キー番号>, <文字列>

文　例

```
KEY 1,"LOAD"
```

解　説

数値はファンクションキーの番号に対応させるため、1～10 までの値でなければなりません。定義できる文字列は、最大 15 文字までの文字およびコントロール文字です(コントロール文字は、CHR$を使って表現します)。一度定義したファンクションキーの内容は、あらたに定義し直すか、一旦電源を切って再び電源を入れるまで変わりませんので注意してください。

参照

CHR$、KEY LIST、KEY ON/OFF

KEY LIST

機能

ファンクションキーの内容を画面に表示します。

書式

KEY LIST

文例

```
KEY LIST
```

解説

[F1]～[F10]までの全てのファンクションキーの内容を画面に表示します。
電源投入時には、ファンクションキーの内容は以下のように設定されています。

F1	"color"
F2	"auto"
F3	"goto"
F4	"list"
F5	"run"+CHR$(13)
F6	"color 15,4,7"+CHR$(13)
F7	"cload"+CHR$(34)（MSX2+では"load"+CHR$(34)）
F8	"cont"+CHR$(13)
F9	"list."+CHR$(13)+CHR$(31)+CHR$(31)
F10	CHR$(12)+"run"+CHR$(13)

参照

KEY、KEY ON/OFF

KEY(n) ON / OFF / STOP

機　能

ファンクションキーが押されたときに特別な処理を行うかどうかを決定します。

書　式

KEY(<キー番号>)ON
KEY(<キー番号>)OFF
KEY(<キー番号>)STOP

文　例

```
10 ON KEY GOSUB 40,60
20 KEY(1) ON:KEY(2)ON
30 GOTO 30
40 PRINT "F1 ガ オサレマシタ"
50 KEY(1) STOP:RETURN
60 PRINT "F2 ガ オサレマシタ"
70 RETURN
```

解　説

この命令は、ON KEY GOSUB によって定義された処理のサブルーチンの実行を許可するか（ON）、禁止するか（OFF）、一時停止するか（STOP）を決定します。

カッコ内の数値は、ファンクションキーのキー番号に対応しています。例えば、KEY(1) ON を実行すると、[F1] を押すごとに、ON KEY GOSUB により定義されている処理のサブルーチンが呼び出されます。サブルーチンが実行がされている間は、KEY(1) STOP が自動的に実行されるので、サブルーチンの実行が終わるまでは、多重にサブルーチンが呼び出されることはありません。KEY(2) OFF を実行すると、[F2] を押しても、用意された処理のサブルーチンは実行されません。KEY(3) STOP を実行すると、[F3] を押しても、用意された処理のサブルーチンは実行されませんが、再び KEY(3) ON を実行することによって、処理が実行されます。この命令は、KEY(3) ON の状態でのみ有効です。

上のプログラムでは、[F1] に関して、一旦 KEY ON していますが(20 行)、その処理のサブルーチンで KEY STOP しているので(50 行)、一度 [F1] キーが押されてしまうと、2 度目からは「F1 ガ オサレマシタ」というメッセージは表示されません。

参　照

ON KEY GOSUB

KEY ON / OFF

機 能

ファンクションキーの内容を画面の下部に表示するかしないかを設定します。

書 式

KEY ON

KEY OFF

文 例

```
KEY ON
KEY OFF
```

解 説

KEY ON を実行すると、ファンクションキーの内容が画面の最下行に表示されます。普通の状態では [F1] ～ [F5] までが、シフトキーを押しているときは [F6] ～ [F10] までが表示されます。

KEY OFF を実行すると、ファンクションキーの表示がとりやめられます。KEY ON / OFF は SET SCREEN 命令により、初期状態を登録することができます。

参 照

KEY、KEY LIST

KILL

Disk

機 能

ファイルを消去します。

書 式

KILL <ファイルスペック>

文 例

```
KILL "A:SAMPLE.BAK"
```

解 説

<ファイルスペック>で指定したファイルを消去します。ファイルスペックの指定には、ワイルドカード（?や＊）を使うことができますが予想外のファイルまで消去してしまわないよう十分注意が必要です。また、ファイルスペック中で、ドライブ名が指定されてない場合は、自動的に現在選択されているドライブ（カレントドライブ）とみなされます。開いているファイルに対して KILL コマンドを実行すると、「File still open」エラーとなります。

LEFT$

機 能

<文字式>の左側から指定された数の文字列を返します。

書 式

LEFT$(<文字式>,<数式>)

文 例

```
10 A$="SUPER HOME COMPUTER"
20 B$= LEFT$(A$,5)
30 PRINT B$:END
```

[表示] SUPER

解 説

与えられた文字列（A$）の左側から、数値（5）で指定された数の文字を取り出します。指定した数が、文字列の総文字数より大きいときは、文字列がそのまま取り出されます。

LEN

機 能

<文字式>で指定された文字列の総文字数を返します。

書 式

LEN(<文字式>)

文 例

```
10 A$="SUPER HOME COMPUTER"
20 PRINT LEN(A$)
30 END
```

表示 19

解 説

LEN はコントロールコードや空白も文字として数えます。

LET

機 能

変数に値を代入します。

書 式

[LET]<変数名>=<式>

文 例

```
10 LET A = 10
20 B = 20
30 PRINT A;B
```

表示 10　20

解 説

代入するものは、文字列でも数値でもかまいません。LET は省略できるので、普通は使いません。

LINE

機 能

グラフィック画面に直線や四角を描きます。

書 式

LINE[[STEP](X1,Y1)]-[STEP](X2,Y2)[,<カラーコード>][,B[F]][,<論理演算子>]

文　例

```
10 COLOR 15,4,7:SCREEN 2
20 LINE(88,56)-(168,136),6,BF
30 LINE(128,76)-(108,116),15
40 LINE -STEP(40,-25),15
50 LINE -STEP(-40,0),15
60 LINE -STEP(40,25),15
70 LINE -(128,76),15
80 GOTO 80
```

解　説

10行でグラフィック画面を指定してから、20行で赤く塗りつぶされた四角を描き、30行から70行でその四角の中に星を描いています。

LINEは、(X1,Y1)で指定した始点と(X2,Y2)で指定した終点とを結ぶ直線を引きます。X軸方向には、SCREEN 2〜5および8以降では255、SCREEN 6と7は511まで、Y軸方向には、SCREEN 2〜4では191、SCREEN 5以降では211までありますので、0からこの値の範囲の点を指定すると、画面に線が描かれます。

色を指定すると、指定した色で線が引かれます。省略した場合は、COLORで指定している前景色が使われます。色と指定する値との対応については、COLORを参照してください。

<カラーコード>の後にBを指定すると、(X1,Y1)から(X2,Y2)を結ぶ線を対角線にした四角が描かれます。BFを指定したときには、四角の中が指定された色で塗りつぶされます(20行参照)。LINEの最後に<論理演算子>を指定すると、LINEの色を次のように操作することができます。ただしこの指定ができるのは、SCREEN 5以降に限られます。

Cを指定色、SCをLINEの下になる色としたときは、

AND	C AND SC
OR	C OR SC
XOR	NOT(C) AND SC
PSET	C
PRESET	NOT(C)
TAND	C AND SC(C=0ならC=SC)
TOR	C OR SC(C=0ならC=SC)
TXOR	NOT(C) AND SC(C=0ならC=SC)
TPSET	C(C=0ならC=SC)
TPRESET	NOT(C) (C=0ならC=SC)

が実際の色となります。

(X1,Y1)を省略して-(X2,Y2)だけが指定されると、最終参照点から(X2,Y2)までを結んだ線が引かれます(70行参照)。最終参照点というのは、線や点などを描くのに1番最後に指定された点のことです。

もう1つのLINEの指定は、STEPを用いたものです。STEPを付けて指定された点は、最終参照点からカッコの中で指定した値の分だけ移動した点を示します。例えば、最終参照点が (100,100) であったとすると、STEP (20,−30) というのは、(120,70) の点を指すことになります。したがって、STEPを用いて引かれる線は、最終参照点とそこからの距離によって指定される点とを結んだものになります (40、50、60行参照)。

80行で80行自身にジャンプしているのは、グラフィック画面では、プログラムの実行が終ると直ちにテキスト画面に移ってしまうからです。これについてはSCREENを参照してください。

参　照

SCREEN、COLOR

LINE INPUT

機　能

キーボードから入力される1行全体の文字列をそのまま文字変数に代入します。

書　式

LINE INPUT["<プロンプト文>";]<文字型変数名>

文　例

```
10 LINE INPUT "デ゛ータ";A$
20 PRINT A$
30 GOTO 10
```

表示 デ゛ータ

解　説

LINE INPUT命令を実行すると、プログラムはキーボードからの入力待ちの状態になります。このとき、疑問符 (?) は表示されません。以降、何かデータを入力すると、リターンキーを押すまでに入力された文字データが画面に表示され、それがそのまま文字型変数に代入されます。キーボードから入力できる文字列の長さは、254文字以内です。

また、LINE INPUT命令では、カンマ (,) や引用符 (") もデータとして入力できます。

なお、INPUT 命令同様、引用符(”)を用いてプロンプト文を指定することができます。

参照

INPUT

LINE INPUT

機能

1行単位のデータをファイルから文字変数へ読み込みます。

書式

LINE INPUT #<ファイル番号>, <文字型変数名>

文例

```
LINE INPUT # 1,A$
```

解説

あらかじめ OPEN 文で INPUT モードを指定して開いておいたファイルからキャリッジリターンコード＋ラインフィードコードで区切られている1行単位(255文字以内)のデータを指定した文字変数に読み込みます。

SAVE 文でアスキー形式のプログラムファイルを作成した場合、一行一行はキャリッジリターンコード(CHR$(13))とラインフィードコード(CHR$(10))とで区切られて書き出されています。また、PRINT #文で作成したファイルには PRINT #文を実行するごとに数値や文字列の並びがキャリッジリターンコード＋ラインフィードコードで区切られて書き出されています。LINE INPUT #文はこれらのファイルからキャリッジリターン＋ラインフィードコードにはさまれているデータを読み込むときに使います。

<ファイル番号>は OPEN 文で指定した番号です。

参照

OPEN、CLOSE、PRINT #、LINE INPUT

LIST、LLIST

機 能

メモリにあるプログラムを画面に表示、プリンタに印刷します。

書 式

[L]LIST[<行番号1>][-[<行番号2>]]

文 例

LIST 100-200

解 説

行番号1と同じか行番号1以上で最も近い番号の行から、行番号2または行番号2以下で最も近い番号の行までのプログラムを画面に表示(LLISTはプリンタに印刷)します。

LIST
プログラムの先頭から最後までが表示されます。
LIST 50
行番号50の行のみ表示されます。
LIST -100
プログラムの先頭から100行目までが表示されます。
LIST 200-
プログラムの200行目から最後までが表示されます。

LISTの機能を中断したいときは、[STOP]キーを押してください。[STOP]キーをもう一度押せば表示を再開します。また、LISTの機能を終了したいときは、[CTRL]+[STOP]キーを押して下さい。
LISTの代わりにLLISTを用いれば、プログラムをプリンタに印刷することができます。

LOAD

機 能

プログラムをメモリ上に読み込みます(ロードする)。

書 式

LOAD<ファイルスペック>[,R]

文　例

LOAD "A:TEST"

解　説

LOADコマンドは<ファイルスペック>で指定したプログラムをメモリ上にロードします。LOAD文を実行すると、LOAD文実行前にメモリ上にあったプログラムは消去されます。また、すべての開いているファイルは閉じられ、変数の値は初期化されます。
<ファイルスペック>のデバイス名を省略すると、カレントドライブからロードします。また、ファイル名はカセットファイルに対して用いる時のみ省略でき、この場合はLOAD "CAS:"とします。ディスクファイルの場合はプログラムのセーブ形式が、バイナリでもアスキーでもかまいませんが、カセットファイルではアスキー形式でセーブしたものに限られます。カセット上のバイナリ形式のプログラム（CSAVEしたもの）はCLOADでのみロードすることができます。
Rオプションをつけるとファイルは開いたままで、プログラムをロード後、ただちに実行を開始します。
LOADは、指定されたファイルを見つけてプログラムのロードを開始するまでは、メモリ上のプログラムを保存しています。

参　照

SAVE、CLOAD

LOC

Disk

機　能

ファイル中の現在の位置を返します。

書　式

LOC（<ファイル番号>）

文　例

IF LOC (1) > 50 THEN STOP

解　説

ランダムファイルを指定した時は、LOC関数は、最後に読んだり書いたりしたレコード番号を値として返します。ただし、オープン直後はLOC関数の値は0になっています。
シーケンシャルファイルを指定したときは、そのファイルがオープンされてから読み出

されたり書き込まれたデータのバイト数を返します（ただし256バイトを1単位とする)。ファイルをシーケンシャル入力モードでオープンした場合は、BASICは最初のセクタを自動的に読むので、そのファイルを読む前でもLOC関数の値は256となります。

LOCATE

機　能

テキスト画面のカーソル位置を指定します。

書　式

LOCATE[<X座標>][,<Y座標>][,<カーソルスイッチ>]

文　例

```
LOCATE 10,12
```

解　説

カーソルを、指定する位置へ移動します。(10,12) なら、画面の左側から11桁目、上から13行目の位置を意味しています(0から指定するため)。LOCATE命令でカーソル位置を指定してから、PRINTやINPUTを実行すると、その位置に文字や?が表示されます。指定された位置が画面の範囲を越えていると、横、縦とも、それを越えない最大の値に設定し直されます。
<カーソルスイッチ>は、0なら入力待ちのときのみカーソルを表示、0以外なら常に表示となります。LOCATEは、テキスト画面および漢字グラフィックモードでのみ使うことができます。

LOF

Disk

機　能

指定されたファイルの大きさを返します。

書　式

LOF（<ファイル番号>)

文 例

```
IF NM%>LOF (1) THEN PRINT "INVALID ENTRY"
```

解 説

指定されたファイルの実際の大きさをバイト単位で返します。

LOG

機 能

<数式>の自然対数を返します。

書 式

LOG(<数式>)

文 例

```
PRINT LOG(10)
```

表示 2.302585092994

LPOS

機 能

プリンタのヘッドの位置を調べます。

書 式

LPOS(<式>)

文 例

```
A = LPOS(0)
```

解 説

現在のプリンタのヘッド位置を調べ、その結果をAに代入します。LPOSによって得られる値は、プリンタバッファ上のプリンタヘッドの位置なので、必ずしも実際の（物理的な）プリンタヘッドの位置とは限りません。カッコ内の数値は形式的なものなので、何であっても構いません。

参 照

LPRINT、WIDTH

LPRINT

機 能

プリンタに文字列や数値を印刷します。

書 式

LPRINT[<式>・・・]

文 例

```
10 A$="ABC":A = 123
20 LPRINT A$,A
30 LPRINT "DEF"
```

解 説

LPRINTに続けて指定された数値（この場合、45）や文字列（DEF）をプリンタに出力します。また、数値変数（A）や文字変数（A$）が指定されると、それらの変数に代入されている数値や文字列がプリンタに出力されます。

LPRINT命令は、出力先がプリンタであるということを除けば、その機能や使い方はPRINT命令と全く同じですので、そちらを参照してください。

参 照

PRINT、PRINT USING、LPRINT USING

LPRINT USING

機 能

文字列や数値を指定した書式でプリンタに印刷します。

書　式

LPRINT USING<書式>；<式>・・・

解　説

LPRINT USINGを実行すると、引用符(")で囲まれた書式にしたがって、指定された文字列や数値が編集され、その結果がプリンタに出力されます。

LPRINT USING命令は、出力先がプリンタであるということを除けば、その機能や使い方はPRINT USING命令と全く同じですので、そちらを参照してください。

参　照

PRINT USING、LPRINT

LSET、RSET　Disk

機　能

ランダム入出力用バッファにデータを転送します（PUTコマンドのための準備)。

書　式

LSET<文字型変数>=<文字式>
RSET<文字型変数>=<文字式>

文　例

```
LSET D$=MKI$(D)
```

解　説

<文字式>の文字が、FIELD文で指定された文字より短い場合は、LSET文では左詰め、RSET文では右詰めでフィールド内を埋め、余分な所は空白で満たされます。逆にFIELD文での割り当てより長い場合は、LSET文、RSET文両方とも文字列の右側が失われます。また、数値データを扱うときは、あらかじめそれらを文字型データに変換しておかねばなりません。それらについては、MKI$、MKS$、MKD$の各関数を参照して下さい。

LSET文、RSET文はFIELD文で定義されていない文字型変数についても、そのフィールド内で左詰めないし右詰めをすることができます。これは、プリント出力の構成に際して利用することができます。

MAXFILES

機 能

使用するファイルの数を定義します。

書 式

MAXFILES=<ファイル数>

文 例

MAXFILES = 3

解 説

OPEN 文で開いて使用するファイルの数を定義します。以降、指定したファイルを同時に開いて使用することができます。定義しないときは、MAXFILES=1 に設定されています。<ファイル数>は 0～15 の範囲を持つ式で、ファイル1個について 267 バイトのファイルコントロールブロック領域が確保されます。MAXFILES 文を実行すると、変数がすべて初期化され、現在オープンしているファイルはクローズされます。また、以前の変数の型宣言などは無効になります。

参 照

CLEAR、OPEN

MERGE

機 能

アスキー形式で保存されているプログラムをメモリ上のプログラムにマージ（混合）します。

書 式

MERGE<ファイルスペック>

文 例

MERGE "CAS:TEST"

解　説

メモリ上のプログラムに<ファイルスペック>で指定したプログラムファイルを混合して1つのプログラムにし、メモリ上に置きます。<ファイルスペック>で指定するファイルはSAVE" ",A文を使ってアスキー形式でセーブされていなければなりません。そうでない場合には、ドライブ名を指定していればエラーとなり、CAS:を指定していればテープが回り続けます。<ファイルスペック>のデバイス名を省略すると、カレントドライブが使用されます。ファイル中のプログラムと、メモリ中のプログラムに同一行番号があった場合には、ファイル中の行でメモリ中の行を置き換えます。MERGEコマンドは実行を終了すると、コマンドレベルにもどります。

参　照

SAVE、LOAD

MID$ ①

機　能

文字列の中から指定された数の文字列を取り出します。

書　式

MID$1(<文字列>, <式1>[, <式2>])

文　例

```
10 A$="SUPER HOME COMPUTER"
20 PRINT MID$(A$,7,4)
30 END
```

表示 HOME

解　説

文字列（A$）の左側から数えて7番目の文字から、4つの文字を取り出します。取り出したい文字数(この場合は4)が省略されたときや、文字列の残りの文字数よりも大きいときは、指定した位置以降のすべての文字列が取り出されます。

参　照

LEFT$、RIGHT$、MID$ ②

MID$ ②

機　能

文字変数の一部を他の文字列と置き換えます。

書　式

MID$(<文字変数>, <式1>[, <式2>])=<文字列>

文　例

```
10 A$="SUPER HOME COMPUTER"
20 MID$(A$,7,4)="ホーム"
30 PRINT A$:END
```

表示 SUPER ホームE COMPUTER

解　説

文字列（A$）の左から数えて7番目から4文字を、右辺の文字列（"ホーム"）と置き換えます。この代入文を実行しても、もとの文字列の長さは変わりません。また、文字数（4）を省略すると<文字列>の文字数と見なされます。

参　照

MID$ ①

MKI$、MKS$、MKD$

Disk

機　能

各数値を内部表現に対応したキャラクタコードに変換します。

書　式

MKI$　(<整数表記>)
MKS$　(<単精度表記>)
MKD$　(<倍精度表記>)

文　例

```
RSET A$= MKI$(123)
```

解　説

ランダムファイルでは、文字データしか扱えないので、数値データは文字型データに変換してディスクに書き込み、読み出す際は逆に文字型データを数値型データに変換します。MKI$は整数値を 2 バイトの文字列に、MKS$は単精度数値を 4 バイトの文字列に、そして MKD$は倍精度数値を 8 バイトの文字列にそれぞれ変換します。
数値から文字への変換は数値が持つ内部表現 (2 進数表現) の値をそのままそれに対応するキャラクタコードにすることによって行われます。この逆の動作をする関数として CVI、CVS、CVD 関数が用意されています。

MOTOR

機　能

カセットレコーダのモーターを ON、OFF します。

書　式

MOTOR[ON または OFF]

文　例

```
MOTOR ON
```

解　説

リモート端子のあるカセットレコーダを接続したときに、モーターの ON、OFF を行って、カセットテープの早送り、巻き戻しができるようにします。MOTOR ON と指定するとモーターが ON になり、MOTOR OFF とすると OFF になります。MOTOR とだけ指定すると、現在モーターが ON なら OFF に、OFF なら ON になります。

NAME　Disk

機　能

ファイルの名前を変更します。

書　式

NAME <ファイルスペック> AS <ファイル名>

文例

NAME "TEST.BAS" AS "TEMP.BAS"

解説

変更しようとするファイルは存在しているものでなければなりません。また、付けようとする名前と同じファイルは存在していてはいけません。ドライブ名を省略すると、カレントドライブ(現在選択されているドライブ)が選択されます。NAME コマンドが実行されると、そのファイルの名前は変更されますが、ファイルの存在する位置や、占有する大きさなどは変化しません。

NEW

機能

メモリにあるプログラムを削除し、すべての変数を初期化します。

書式

NEW

文例

NEW

解説

NEW を実行すると、メモリにあるプログラムはすべて削除され、定義されている変数はすべて初期化され、開かれているファイルはすべて閉じられます。

参照

DELETE、ERASE

OCT$

機能

＜数式＞の数値を8進数の文字列に変換します。

書　式

OCT$(<数式>)

文　例

```
PRINT OCT$(100)
```

表示 144

参　照

BIN$、HEX$

ON ERROR GOTO

機　能

エラーが起こったときにジャンプする行を指定します。

書　式

ON ERROR GOTO<行番号>

文　例

```
10 ON ERROR GOTO 60
20 A$=INKEY$
30 IF A$="Y" THEN 80
40 A%=A%+100
50 GOTO 20
60 PRINT "ショリ":A%=0
70 RESUME
80 ON ERROR GOTO 0
```

解　説

10行で、エラーが起こったときに60行にジャンプするように指定してあるので、40行のA%が32767を越えて「Overflow」エラーが起きると60行にジャンプし、A%に0を代入します。
あらかじめ、エラーを処理する部分をプログラム中に入力しておき、プログラムの最初にON ERROR GOTOでその部分の最初の行を指定しておけば、エラーが起きたときに、エラー処理用の指定が実行されます。エラー処理用の指定の最後にはRESUME命令を置くと、エラー処理を終了させ、元のプログラムを再開させることができます（70

行参照)。

ON ERROR GOTO 命令は、他の ON ERROR GOTO 命令、RUN、CLEAR、NEW が実行されるまで有効ですので、そのままプログラムを終了すると、ダイレクトモードでタイプミスなどをしても、再びエラー処理用の行へジャンプし、プログラムが実行されてしまいます。したがってプログラムの終了直前には、ON ERROR GOTO の指定を無効にする、ON ERROR GOTO 0 を実行するようにしてください (80 行参照)。

参　照

RESUME、ERL、ERR、ERROR

ON GOTO、ON GOSUB

機　能

<式>で指定されたいずれかの行にジャンプします。

書　式

ON<式>GOSUB<行番号>[,<行番号>・・・]
ON<式>GOTO<行番号>[,<行番号>・・・]

文 例 1

```
10 INPUT "1 カ 2 ヲ オシテクダ゛ サイ";A
20 ON A GOTO 100,200
30 GOTO 10
100 PRINT "100 キ゛ ョウニシ゛ ャンフ゜ シマシタ":END
200 PRINT "200 キ゛ ョウニシ゛ ャンフ゜ シマシタ":END
```

文 例 2

```
10 INPUT "1 カ 2 カ オシテクダ゛ サイ";A
20 ON A GOSUB 100,200
30 GOTO 10
100 PRINT "100 キ゛ ョウニシ゛ ャンフ゜ シマシタ":RETURN
200 PRINT "200 キ゛ ョウニシ゛ ャンフ゜ シマシタ":RETURN
```

解　説

ON と GOTO、あるいは ON と GOSUB の間の値 (A) に応じて、GOTO、GOSUB の後ろで指定した行にジャンプします。ON　GOTO は指定した行にジャンプし、ON

GOSUBは指定した行で始まるサブルーチンにジャンプします（GOSUB参照）。
GOTO、GOSUBの後ろの指定は、左から1、2、3・・・という数に対応していて、例えば、上の例では、1が押されれば100行に、2が押されれば200行にジャンプします。
対応する行番号がないとき（上の例では0や3以上の数値を入力した場合）は、次の命令が実行されます。負の数が指定されたときは、Illegal function callエラーとなります。

参　照

GOTO、GOSUB

ON INTERVAL GOSUB

機　能

指定した時間ごとにサブルーチンの呼び出しを行うように設定します。

書　式

ON INTERVAL=<時間>GOSUB<行番号>

文　例

```
10 ON INTERVAL = 600 GOSUB 40
20 INTERVAL ON
30 GOTO 30
40 PRINT"*";
50 RETURN
```

解　説

INTERVALで指定する時間は、1から65535までの数値で、単位は1/60秒です。ON INTERVAL GOSUB命令は、設定を行うだけです。設定後にINTERVAL ONを指定して初めて、指定した時間が経つとサブルーチンにジャンプします。サブルーチンから戻るには、RETURN命令を使います。また、RETURNに戻り先の行番号を指定することもできます。
RETURN命令が実行されるまでは、INTERVAL STOPが実行されたのと同じ状態になっているので、多重にインターバル割り込みがかかることはありません。
上のプログラムは10秒ごとに「＊」を表示するものです。10行で、ON INTERVALで指定した時間（600）ごとに、GOSUBで指定された行（40行）にジャンプするように設定しています。

参照

INTERVAL ON/OFF/STOP

ON KEY GOSUB

機能

ファンクションキーが押されたときにジャンプする行を指定します。

書式

ON KEY GOSUB[<行番号>][,<行番号>・・・]

文例

```
10 ON KEY GOSUB 40,60
20 KEY(1)ON:KEY(2)ON
30 GOTO 30
40 PRINT "F1 カ゛ オサレマシタ"
50 RETURN
60 PRINT "F2 カ゛ オサレマシタ"
70 RETURN
```

解説

ファンクションキーが押されたときに、呼び出すサブルーチンの行番号を指定します。ON KEY GOSUB に続く行番号（40 と 60）が、実際に処理を行うサブルーチンの先頭行で、指定されたファンクションキー（[F1] と [F2]）が押されたときには、プログラムの実行はここに移ります。

行番号の並びの順序は、ファンクションキーのキー番号に対応しています。ファンクションキーは [F1] から [F10] まであるので、最大 10 個の行番号が並べられます。

呼び出されたサブルーチンは、GOSUB 命令同様、RETURN 命令によってジャンプした次の行に戻ります。また、RETURN に戻り先の行番号を指定することもできます。ただし、この命令は、ファンクションキーが押されたときの処理の定義を行うだけで、これだけでは処理は実行されません。上のプログラムでも、20 行が実行されて、初めて 40 行と 60 行が実行されています。詳しくは、KEY(n) ON/OFF/STOP を参照してください。

参照

KEY(n) ON/OFF/STOP

ON SPRITE GOSUB

機能

スプライトパターンが重なったときにジャンプする行を指定します。

書式

ON SPRITE GOSUB<行番号>

文例

```
10 SCREEN 1,1
20 SPRITE$(0)=STRING$(8,CHR$(&B11111111))
30 ON SPRITE GOSUB 90
40 PUT SPRITE 0,(100,100),1,0
50 SPRITE ON
60 X=255
70 PUT SPRITE 1,(X,100),6,0
80 X=X-1:GOTO 70
90 REM SUB
100 X=255
110 RETURN
```

解説

30行で、スプライトのパターンが重なったときには、90行にジャンプするように指定しています。

ON SPRITE GOSUBを実行しておき、SPRITE ONを実行すると、画面上でスプライトパターンが重なったときに、ON SPRITE GOSUBで指定した行（90）へジャンプします。ジャンプした行以降で、スプライトが重なったときの処理を行ったあとは、RETURNで元の行へ戻ります。また、RETURNに戻り先の行番号を指定することもできます。

なお、SCREEN 4以降のスプライトで特殊なカラーコードを持つものは衝突しても割り込みを起こしません。詳しくは、COLOR SPRITE文を参照してください。

参 照

SPRITE ON / OFF / STOP

ON STOP GOSUB

機 能

CTRL + STOP キーが押されたときにジャンプする行を指定します。

書 式

ON STOP GOSUB<行番号>

文 例

```
ON STOP GOSUB 100
```

解 説

CTRL + STOP キーが押されると、プログラムの実行は100行以降に移ります。ON STOP GOSUBは、CTRL + STOP キーが押されたときに、分岐するサブルーチンの行番号を指定するものです。ON KEY GOSUBに続く行番号（この場合、100）が、実際に処理を行うサブルーチンの先頭行になります。呼び出されたサブルーチンは、GOSUB命令同様、RETURN命令によってジャンプした次の行に戻ります。また、RETURNに戻り先の行番号を指定することもできます。ON KEY GOSUBの説明を参照して下さい。

参 照

ON KEY GOSUB、STOP ON / OFF / STOP

ON STRIG GOSUB

機 能

ジョイスティックのトリガボタンが押されたときにジャンプする行を指定します。

書 式

ON STRIG GOSUB<行番号>[, <行番号>・・・]

文 例

ON STRIG GOSUB,100,200

解 説

ポート1に接続されているジョイステックのトリガボタン1が押されたらプログラムの実行を100行以降に移し、ポート2に接続されているジョイステックのトリガボタン1が押されたらプログラムの実行を200行以降に移します。
この命令は、ジョイスティックのトリガボタンが押されたときに、分岐するサブルーチンの行番号を指定するものです。ON STRIG GOSUBに続く行番号（この場合、100と200)が、実際に処理を行うサブルーチンの先頭行で、指定されたトリガボタンが押されたときには、プログラムの実行はここに移ります。
行番号は最大5個まで指定することができ、左側から順に以下のトリガボタンと対応しています。

行番号	意　味
1番目	キーボードのスペースキー
2番目	ポート1に接続されているジョイスティックのトリガボタン1
3番目	ポート2に接続されているジョイスティックのトリガボタン1
4番目	ポート1に接続されているジョイスティックのトリガボタン2
5番目	ポート2に接続されているジョイスティックのトリガボタン2

ただし、ジョイスティックによっては、トリガボタン2のないものもあります。
呼び出されたサブルーチンは、GOSUB命令同様、RETURN命令によってジャンプした次の行に戻ります。また、RETURNに戻り先の行番号を指定することもできます。ただし、この命令は、ジョイスティックのトリガボタンが押されたときの処理の定義を行うだけで、これだけでは処理は実行されません。STRIG(n) ON/OFF/STOPを参照して下さい。

参 照

STRIG(n) ON/OFF/STOP、ON KEY GOSUB

OPEN

機 能

ファイルを開きます。

書　式

OPEN <ファイルスペック>[FOR<モード>]AS #<ファイル番号> [LEN =<レコード長>]

文　例

OPEN "SAMPLE.DAT" FOR OUTPUT AS # 1

解　説

OPEN 文は、ファイルを使用するにあたってバッファエリアを割り当て、アクセスモードを決定します。一度ファイルをオープンすると、以後は、<ファイル番号>でデータのやりとりを行います。以下の命令や関数を使用する前には、あらかじめ OPEN 文を実行しておくことが必要です。

EOF	FIELD
GET	INPUT #
INPUT$ (n,# m)	LINE INPUT #
LOC	LOF
PRINT #	PRINT # USING
PUT	VARPTR (# n)

<ファイルスペック>でドライブ名を省略したときは、現在のディスクドライブが使用されます。

<モード>はファイルに対するアクセスのモードを決定するもので、次の4種があります。

FOR INPUT	既存のシーケンシャルファイルから入力を行う。
FOR OUTPUT	新しくシーケンシャルファイルを作り出力を行う。
FOR APPEND	既存のシーケンシャルファイルに新たなデータを追加出力する。
<モード>省略	ランダムファイルに対して入出力を行う。

<ファイル番号>は、1から15までの値を用いることができますが、MAXFILES 文で指定した数を越えて指定することはできません。また、既にオープンされているファイルが使用している番号を用いることはできません。

シーケンシャル入力モードおよび追加モードでは、指定されたファイルが存在しないと、「File not found」エラーとなります。シーケンシャル出力モードでは、常に指定された名前のファイルを新しく作り、同一名のファイルがあった場合にはそのファイルは削除されます。ランダム入出力モードでファイルが存在しない場合には新たに作られます。<レコード長>は、ランダム入出力モードでファイルをオープンしたときの、レコード長を設定するものです。これは1から256の範囲で、省略した場合は256に設定されます。

シーケンシャル入力、あるいはランダム入出力モードを指定する場合には、1つのファイルを複数のOPEN文により開くことができます。この場合、異なるファイル番号により指定することになりますが、シーケンシャル出力モードを指定する場合には、1つのファイルを複数のOPEN文により開くことはできません。

■ファイルの指定

OPENに続く＜ファイルスペック＞で、そのファイルを操作する場所とファイル名を指定します。ディスクファイル以外では次の7つがあります。

カセットレコーダ	"CAS:ファイル名（6文字以内）"と指定する
メモリディスク	"MEM:ファイル名"と指定する
ディスプレイのテキスト画面	"CRT:"と指定する
ディスプレイのグラフィック画面	"GRP:"と指定する
プリンタ	"LPT:"と指定する
コンソールデバイス	"CON"と指定する
ヌルデバイス	"NUL"と指定する

コンソールデバイスはキーボードからの読み込み、画面への書き込みに使用します。例えば、

```
COPY "TEST.DOC" TO "CON"
```

とすると、TEST.DOCの内容を画面に表示します。
ヌルデバイスは実際には何もしません。出力文字は無視され、入力は常にエンドオブファイル（CTRL + Z）となります。
ファイル名が必要なのは、ディスクファイルおよび、CAS:やMEM:だけです。

参照

CLOSE、INPUT #、PRINT #、MAXFILES、INPUT$、EOF

OUT

機能

指定した出力ポートにデータを送ります。

書 式

OUT<ポート番号>,<式>

文 例

```
OUT &H22,&H80
```

解 説

&H22のポート番号で指定された出力ポートへ、&H80というデータを送ります。ポート番号に関しては、I/Oマップを参照してください。

OUTは、ハードウエアと非常に密着した命令で、各々のMSXで異なる動作をする場合があります。MSXの互換性を維持するため、広く公表するプログラムには使用しないでください。

参 照

INP

PAD

機 能

タッチパネルの状態を調べます。

書 式

PAD(<数式>)

文 例

```
X = PAD(1)
```

解 説

ポート1に接続されたタッチパネルが押されていたら、その点のX座標を変数Xに代入します。

PADに続くカッコ内には、0から19まで値が入ります。PAD命令を実行すると、指定によって、以下の結果が返されます。

数値	意　味	出される値
0、4	タブレットのデータが有効かどうかを調べる	有効=−1
		無効=0
1、5	タブレットが押されている点のX座標を調べる	X座標
2、6	タブレットが押されている点のY座標を調べる	Y座標
3、7	タブレット上のスイッチが押されているかどうか調べる	押されている=−1
		押されていない=0
*8	ライトペンのデータが有効かどうかを調べる	有効=−1
		無効=0
*9	ライトペンのX座標を調べる	X座標
*10	ライトペンのY座標を調べる	Y座標
*11	ライトペンのスイッチが押されているかどうかを調べる	押されている=−1
		押されていない=0
*12、*16	マウスまたはトラックボールのデータを与える	常に−1
*13、*17	マウスまたはトラックボールのX座標を調べる	X座標
*14、*18	マウスまたはトラックボールのY座標を調べる	Y座標
*15、*19	マウスまたはトラックボールのデータを与える	常に0

注　意

0〜3または12〜15はポート1に接続された装置を、4〜7または16〜19はポート2に接続された装置を調べます。ただし、*のついた装置（8〜19）はMSX BASIC ver 2.0以降で有効です。

どの周辺装置も座標を調べる前にはそのデータが有効かどうかを確認しなければなりません。それぞれPAD(0)または(4)、PAD(8)、PAD(12)またはPAD(16)を実行して、その値が−1であったら他のデータを調べて下さい。

参　照

STRIG、PDL

PAINT

機　能

グラフィック画面の、指定された境界色でかこまれた部分を指定された色で塗りつぶします。

書　式

PAINT[STEP](X,Y)[,<領域色>][,<境界色>]

文例

```
10 SCREEN 3
20 CIRCLE(128,96),50,11
30 PAINT(128,96),6,11
40 GOTO 40
```

解説

(128,96) を中心点にして、黄色で描かれた円の中を赤で塗ります。
PAINT の指定は次の形で行います。

PAINT(X,Y)、領域色、境界色

PAINT は、(X,Y) で指定した位置を含む境界色で囲まれた領域を、領域色で塗りつぶします。色の指定については、COLOR を参照してください。
(X,Y)の指定は STEP 使って行うこともできます。STEP については、LINE を参照してください。
境界色は SCREEN 3 および 5 以上でだけ使えます。SCREEN 2 や 4 では、領域色と同じ色で囲まれた領域が、領域色で指定した色で塗られます。例えば、上の例を SCREEN 2 で行うと次のようになります。

```
10 SCREEN 2
20 CIRCLE(128,96),50,11
30 PAINT(128,96),11
40 GOTO 40
```

領域色の指定を省略すると、COLOR で指定している前景色が使われます。境界色の指定を省略すると、領域色で指定したものと同じ色が使われます。PAINT を実行したときに、指定した境界色 (SCREEN 2 の場合には領域色) で囲まれた領域が無いと、画面一面が指定した領域色で塗りつぶされます。

参照

SCREEN、COLOR

PDL

機能

パドルの状態を調べます。

書　式

PDL(<パドル番号>)

文　例

```
X = PDL(2)
```

解　説

例ではポート2に接続されたパドルの状態を調べ、その結果を変数Xに代入します。
PDLに続くカッコ内には、1から12までの値が入ります。数値が奇数(1、3、5、7、9、11)ならば、ポート1に接続されたパドルが指定され、偶数(2、4、6、8、10、12)ならば、ポート2に接続されたパドルが指定されます。
PDL命令を実行すると、パドルの状態がその回転の度合によって、0〜255の間の値で得られます。

参　照

PAD、STRIG

PEEK

機　能

メモリ上の指定された番地の内容を読み出します。

書　式

PEEK(<番地>)

文　例

```
A = PEEK(&HA000)
```

解　説

例ではカッコ内で指定された番地(&HA000)のメモリの内容が、数値としてAに代入されます。読み込み番地は、−32768〜65535の範囲の数値で指定します(数値が負の場合は、その値に65536を加えれば実際の番地を知ることができます)。読み込まれるデータは、0〜255の値を持つ数値です。

参　照

POKE、メモリマップ

PLAY ①

機能

音楽を演奏します。

書式

PLAY<文字式1>[,<文字式2>][,<文字式3>]

文例

```
10 PLAY "O4L4T180V8"
20 PLAY "GAGAGGE2"
30 PLAY "EFEFEECR"
40 PLAY "E2CDE2CD"
50 PLAY "EFGGDEDR"
60 PLAY "GAGAGGE2"
70 PLAY "EFEFEECR"
80 PLAY "A2GEGGEC"
90 PLAY "EEDDC2"
```

解説

指定された記号にしたがって、音楽の演奏をします。10行では、演奏のためのテンポ(はやさ) や音量、音階の音域などを指定しています。
文字式は、音楽を演奏するための記号を組み合わせたもので引用符で囲みます。文字変数で指定するときは、その文字変数に演奏用の記号を代入しておきます("X文字変数;"と指定しても、単に文字変数だけを指定しても、結果は変わりません)。記号には、音の高さ、音の長さ、速さ(テンポ)、音の大きさ、音色の指定の5種類があります。
PLAYでは、3つのチャンネルを使うことができ、最大3重和音まで出すことができます。文字式1、文字式2、文字式3はそれぞれチャンネル1、2、3に対応しています。文字式に何も指定しなければ、そのチャンネルからは音がでません。
以下で、各記号についての解説を行います。

例 PLAY "C","E","G"

■**音の高さ**

A〜G　　音の高さを音階名で表現する。A〜Gの後ろに#、+を付けると半音上がり、−を付けると半音下がる。

O1〜8　　音階の音域を1〜8の数値で指定する。初期値はO4。例えば、O5は

O4より1オクターブ上の音階を指す。

N0〜96　音の高さを0〜96の数値で表現する。N1はO1のC#と同じ高さ、N96はO8のBと同じ。N0はR（休符）と同じ。

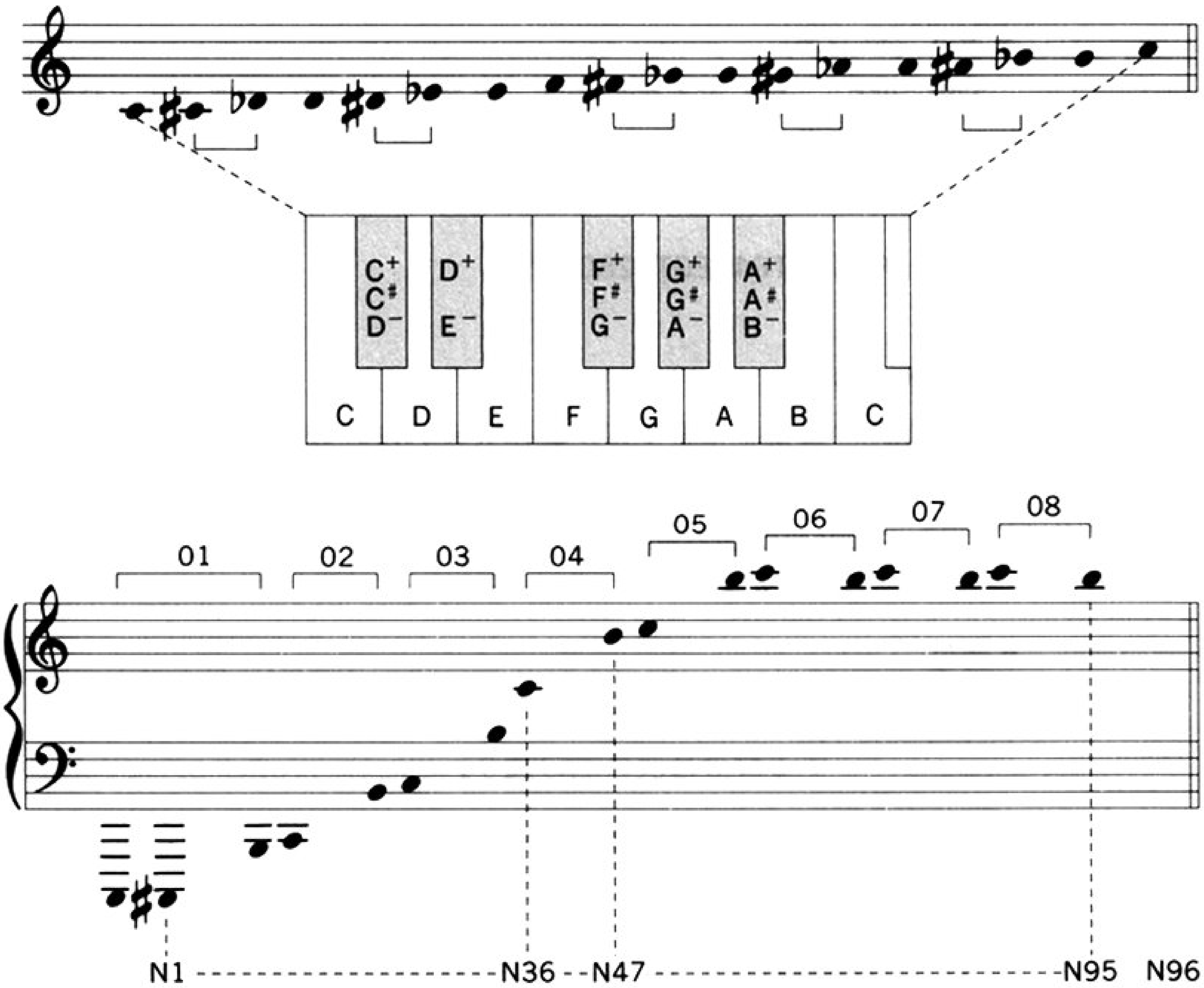


図 2.13　音の高さ

例	PLAY "CDE" PLAY "O3CDEO4CDEO5CDE" PLAY "N36N38N40"

■ **音の長さ**

L1〜64　音の長さを1/数値音にする。L1は全音符(4拍)、L2は2分音符(2拍)、L64は64分音符(1/16拍)。1音だけ長さを変えるときは、音階名のうしろに数値だけをつける。

.（ピリオド）　音階名、休符の後ろに付けて音の長さを1.5倍する。

R数値　休符を指定する。数値の指定の方法は、Lのときと同じ。数値を省略するとR4とみなされる。

例　PLAY "L2CEGL4CEGL8CEG"
PLAY "L4CEGCE8G"
PLAY "L4CD.E..F..."
PLAY "CR4DR2E"

■速さ（テンポ）

T32〜255　テンポを指定する。数値で1分間に演奏する4分音符の数を指定する。初期値はT120

例　PLAY "T64CEGT128CEG"

■音の大きさ

V0〜15　音量を指定する。数が大きいほど音も大きくなる。

例　PLAY "V2CV4DV6EV8FV10G"

■音色

S0〜15　音量変化の波を指定する。Vとは一緒に指定できない。数値に対応する波形は次の図の通り。

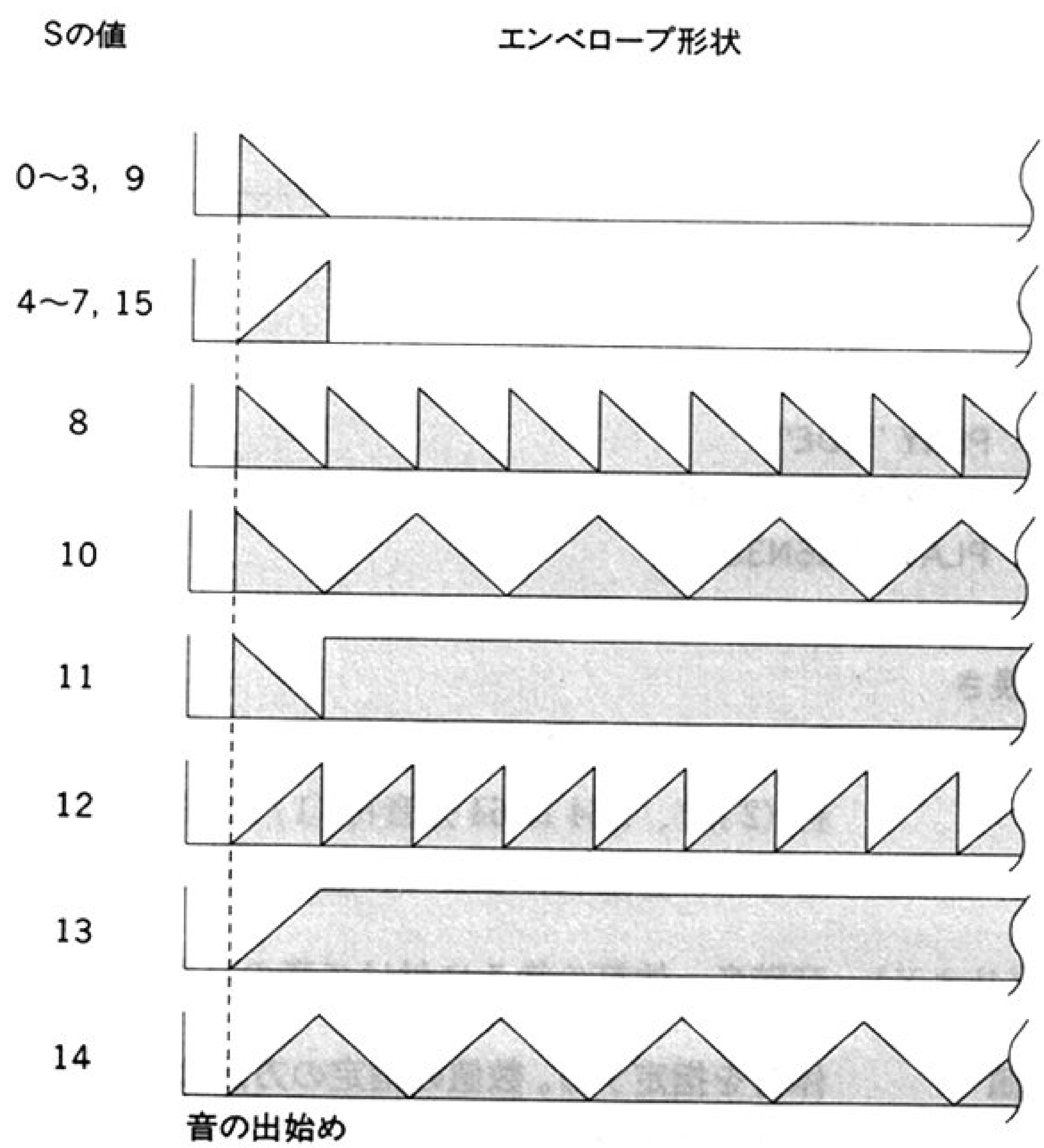


図2.14　エンベロープ形状

例

```
10 FOR J = 8 TO 14
20 PLAY "S = J;M1000CDE"
30 NEXT
```

M1〜65535　音量変化の波の周期を指定する.

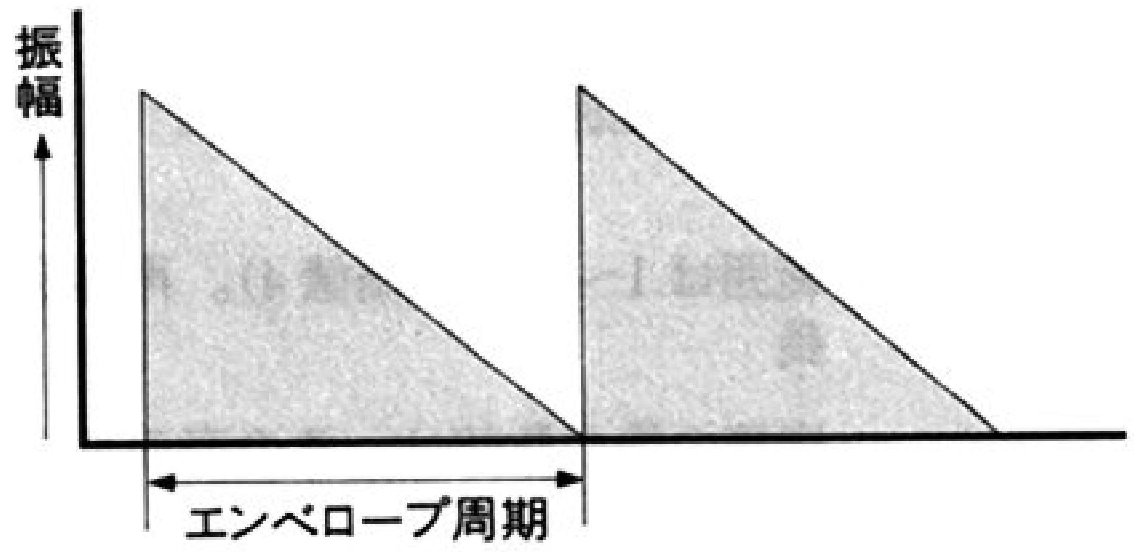


図 2.15　エンベロープ周期

例

```
10 FOR J = 100 TO 3000 STEP 100
20 PLAY "S8M = J;CDE"
30 NEXT
```

O、L、T、V の指定は、次に別の値で指定が行われるまで有効です。また V を指定すると、そのチャンネルの S や M の指定はキャンセルされます。

■ **数値変数での指定**

なお、次の形式で数値変数も指定できます。

＜記号＞=数値変数；

例

```
10 FOR J = 1 TO 64
20 PLAY "L = J;CDE"
30 NEXT
```

表 2.11　マクロ一覧

	記号	意　味	備　考	例
音程	A～G	ラ～ソ	音階名	PLAY"CDE"
	#、+	シャープ	半音上げる	PLAY"CD #"
	-	フラット	半音下げる	PLAY"CD-"
	O 数値	オクターブ	音域指定。範囲は 1～8（省略値 4）	PLAY"O5C"
	N 数値	音程	数値の範囲は 0～96 で半音づつ上がる。0 は休符	PLAY"N36"
音長	L 数値	音符長	範囲は 1～64（省略値 4）。再指定までの音符長	PLAY"L4C"
	数値	音符長	音符の後の数値は、その音の長さだけ指定	PLAY"C32"
	.	符点	A～G、N、R の後に付けて長さを 1.5 倍する	PLAY"C4."
	R 数値	休符長	L と同じ指定だが休符は別に管理される	PLAY"CR8"
速さ	T 数値	テンポ	範囲は 32～255（省略値 120）	PLAY"T64"
音量	V 数値	音量	範囲は 0～15	PLAY"V8C"
音色	S 数値	波形	範囲は 0～15。V と一緒には指定できない	PLAY"S10"
	M 数値	周波数	範囲は 1～65535。S と共に用いる	PLAY"M10"

PLAY ②

機　能

音楽を演奏中かどうかを調べる関数です。

書　式

PLAY(<ボイスチャンネル番号>)

文　例

```
10 A$="O4L4T120V4"
20 PLAY A$,A$,A$
30 PLAY "C","E","G"
40 PLAY "C","F","A"
50 PLAY "B","A","G"
60 IF PLAY(0)=0 THEN 30
70 GOTO 60
```

解説

PLAY（数値）は、各チャンネルで、指定した音楽を演奏中かどうかを調べます。演奏中であると−1、そうでなければ0が与えられます。

上のプログラムはまず10行で音階の音域、音の長さなどの設定を行って20行でそれを3つのチャンネルで実行させています。30行から50行で音を出し、60行で演奏が終わったかどうかを調べて、終わっていたら30行にジャンプします。70行で60行にジャンプしているのは、PLAYの値が0になるのを待つためです。プログラムでは、50行に続いて60行がすぐ実行されるのですが、演奏は続いているので、1度だけ60行で調べてもPLAYの値は−1で、30行へのジャンプが行われなくなってしまうからです。

PLAYの指定は次の形で行います。

PLAY（チャンネル）

チャンネルは0〜3の値で指定します。

0	チャンネル1、2、3（どれかが演奏中であれば−1となる）
1	チャンネル1
2	チャンネル2
3	チャンネル3

POINT

機能

グラフィック画面上の指定した点の色を調べます。

書式

POINT(X,Y)

文例

```
10 R = RND(-TIME)
20 C = INT(RND(1) * 15+1)
30 COLOR ,C,C
40 SCREEN 2
50 A = POINT(128,96)
60 SCREEN 1
70 PRINT A
80 END
```

解　説

POINTは、指定した座標の色を調べて、カラー番号で返します。ただし、スプライトの色は調べることができません。
上のプログラムは乱数を使って画面の色を変え、50行でその色を調べています。そのあと60行で画面を切り替え、70行でカラー番号を表示しています。

参　照

COLOR

POKE

機　能

メモリ上の指定された番地にデータを書き込みます。

書　式

POKE<番地>,<データ>

文　例

```
POKE &HA000,57
```

解　説

書き込み番地は、−32768〜65535の範囲の数値で指定します(数値が負の場合は、その値に65536を加えれば実際の番地を知ることができます)。書き込むデータは、0〜255の値を持つ数値です。
この命令は、現在のメモリ内容を書きかえてしまうため、不用意に使うとエラーやシステムの暴走の原因になることがあります。使うときには、メモリマップなどで使用できる領域かどうかを確認してください。
上の例では、メモリ上の&HA000番地に57というデータを書き込みます。

参　照

CLEAR、PEEK、メモリマップ

POS

機　能

テキスト画面上のカーソルの水平位置を調べます。

書　式

POS(<式>)

文　例

```
10 FOR I = 10 TO 99
20 A = POS(0)
30 IF A>12 THEN PRINT
40 PRINT I;
50 NEXT
```

解　説

POSはカーソルが左から何桁目にあるかを、一番左の位置を0とした値で調べます。引き数は仮のものなので、どんな数値でもかまいません。POSは、テキスト画面でのみ使うことができます。

上のプログラムは、単に10から99までの値を画面に表示するものですが、20行でPOSを使ってカーソルの水平位置を調べ、30行で、その値が12を越えたら改行を行うように指示しています。

参　照

CSRLIN

PRINT

機　能

テキスト画面に文字列や数値を表示します。

書　式

PRINT[<式>・・・]

文　例

```
10 A$="ABC":A = 123
20 PRINT A$,A
30 PRINT A$;A
40 PRINT
50 PRINT 456 "DEF";
60 PRINT "IIII"
```

解　説

PRINT に続けて数値 (456) や文字列 (DEF、IIII) を指定すると、その数値、文字列が画面に表示されます。また、数値変数 (A) や文字変数 (A$) が指定されると、それらの変数に代入されている数値 (123) や文字列 (ABC) が画面に表示されます。PRINT だけを指定すると (40 行) 改行が行われます。

数値を表示する場合、その後ろに必ず空白が1つ入ります。また、数値の前には符号用の桁が用意されて、プラスのときには空白、マイナスのときにはマイナス符号が表示されます。

複数のデータを表示する場合、それらの区切りとして、カンマ (,)、セミコロン (;)、空白が使えます。カンマを使うと(20 行)、PRINT の後ろの奇数番目のデータは画面の一番左の列に表示され、偶数番目のデータは 15 桁目に表示されます。ただし上の例の場合、2 番目のデータが数値なので見かけ上は 16 桁目に表示されます。

区切りにセミコロンあるいは空白を使うと(30 行、50 行)、PRINT の後ろのデータがつながって画面に表示されます。一番最後のデータの後ろにセミコロンを置くと、次の PRINT 文のデータが前のデータにつながって表示されます (50 行と 60 行)。

PRINT の省略形として? (疑問符) を使うことができます。

PRINT

機　能

シーケンシャルファイルにデータを出力します。

書　式

PRINT #<ファイル番号>, [<式>･･･]

文　例

```
PRINT # 1,A$
```

解　説

＜ファイル番号＞は、そのシーケンシャルファイルが出力モードとしてオープンされたときに指定された番号です。

＜式＞は、ファイルに書き込まれる数値式または文字式です。

PRINT #文は画面に出力するのと同じものをファイルに出力します。そのため、データを読むときに正しく入力できるように、書き込むデータを適切に区切る必要があります。

シーケンシャルファイルに対するデータの出力にも、PRINT # USING 文が用意してあります。カンマ(,)は省略できません。

参　照

OPEN、CLOSE、INPUT #、LINE INPUT #、INPUT$、PRINT # USING、PRINT、LPRINT

PRINT USING

機　能

文字列や数値を指定した書式でテキスト画面に表示します。

書　式

PRINT USING＜書式＞；＜式＞・・・

文　例

```
10 A$="ABCDE":B$="ZYXWV"
20 A = 123.45:B =−98.765
30 'モ゛ シレツ
40 PRINT USING "!";A$,B$
50 PRINT USING "& &";A$
60 PRINT USING "@ MNL @";A$,B$
70 'スウチ
80 PRINT USING "+####. ####";A,B
90 PRINT USING "####. ###-";B
100 PRINT USING "**##. ###";B
110 PRINT USING "¥¥##. ###";B
120 PRINT USING "**##. ###";A
130 PRINT USING "####,. ###";A
140 PRINT USING "####. ###^^^^" ; A
```

```
150 PRINT USING "##";A
160 PRINT USING "ゴウケイ###.###"; B
```

表示

```
AZ
ABC
ABCDE MNL ZYXWV
 +123.4500   -98.7650
   98.765-
*-98.765
-¥98.765
*123.450
   123.450
 123.450E+00
%123
ゴウケイ -98.765
```

解　説

PRINT USINGの後ろに指定した、書式文字列でセミコロン（；）以下の文字列や数値を編集して表示します。

複数のデータを表示するときには、カンマ（,）かセミコロンで区切ります。カンマで区切ってもセミコロンで区切っても同じです。

書式には、文字列を編集するものと数値を編集するものとがあります。以下に書式として使う記号を説明します。

■ **文字列を編集するもの**

!	文字列の左側1字を表示する。
&空白&	文字列の左側から、指定した空白の数+2文字を表示する。
@	@を指定した文字列で置き換える。

■ **数値を編集するもの**

#	数値を#で指定した桁数だけ表示する。
.	小数点の位置を指定する。
+ −	数値に正負符号を付ける。書式の左端に付けると、数値の前に、右端に付けると数値の後ろに正負符号を付ける。
**	書式の左端を**にしておくと、数値の整数部の桁数が書式の指定より小さいときに、小さい分だけ*が表示される。
¥¥	書式の左端を¥¥にしておくと、数値の直前に¥が付けられる。¥¥は指数形式の書式指定をしているときには使えない。
**¥	書式の左端を**¥にしておくと、数値の直前に¥が付けられ、整数部の桁数が書式の指定より小さいときに、小さい分だけ*が表示さ

	れる。＊＊¥は指数形式の書式指定をしているときには使えない。
,	,を整数部の#と#の間、あるいは小数点の左側に置いたときには、整数部が3桁ごとにカンマで区切られて表示される。小数点より右に置いたときには、数値の右にカンマが付けられる。カンマ（,）は指数形式の書式指定をしているときには使えない。
^^^^	^^^^を#の後ろに付けると、数値が指数形式で表示される。書式に＋または－を指定していない場合、数値が正なら数値の前に空白が1つ、負なら数値の前に－が表示される。
%	表示しようとする編集済みの数値が書式で指定した桁数より大きいと、数値の前に%が付けられる。

書式の中にこれらの記号以外の文字を置くと文字の位置に応じて数値の前や後ろにその文字が表示されます（160行参照）。

PRINT # USING

機　能

指定されたファイルに指定した書式でデータを書き出します。

書　式

PRINT # <ファイル番号>,USING<書式>；<式>・・・

文　例

```
10 OPEN "CRT:" FOR OUTPUT AS # 1
20 A = 123:B =-45
30 A$="SUPER"
40 PRINT # 1,USING"####. ##-";A;B
50 PRINT # 1,USING"@  MSX";A$
60 CLOSE
```

解　説

PRINT # USINGを実行すると、書式文字列にしたがって、指定された文字列や数値が編集され、その結果がファイルに書き出されます。PRINT # USINGは、データを指定された書式にしたがって出力することを除けば、その機能はPRINT #と同じです。また、書式を使った編集方法は、PRINT USINGと同じですので、そちらを参照してください。

参照

PRINT #、PRINT USING

PSET、PRESET

機能

グラフィック画面上に点を描いたり (PSET)、描いてある点を消したり (PRESET) します。

書式

PSET[STEP](X,Y)[,<カラーコード>][,<論理演算子>]
PRESET[STEP](X,Y)[,<カラーコード>][,<論理演算子>]

文例

```
10 SCREEN 3
20 X = INT(RND(1) *255)
30 Y = INT(RND(1) *192)
40 C = INT(RND(1) *15+1)
50 PSET(X,Y),C
60 FOR I = 0 TO 100:NEXT
70 PRESET(X,Y)
80 GOTO 20
```

解説

PSET は、(X,Y) で指定した位置に C で指定された色の点を描きます (色の指定を省略すると、COLOR 命令で指定した前景色が使われます)。
PRESET で色の指定を省略すると、指定した位置に描かれている点を消します(このとき背景色以外の色を指定すると、その色で点が描かれてしまいますので注意してください)。これらの指定には STEP を使うこともできます。STEP については LINE を参照してください。また、色コードの後には<論理演算子>を指定できる場合があります。これについても LINE を参照してください。

参　照

COLOR、LINE

PUT

Disk

機　能

ランダム入出力バッファ中のデータをランダムファイルへ出力します。

書　式

PUT [#]<ファイル番号>[,<レコード番号>]

文　例

```
PUT #1,1
```

解　説

<ファイル番号>で指定されたファイルに対するバッファの内容を書き出します。指定されたファイルは、ランダムアクセスのモードでオープンされていなければなりません。<レコード番号>は、1から4,294,967,295までが有効で、バッファ中のデータが指定されたレコードに書き込まれます。<レコード番号>が省略された場合には、直前のGET文、PUT文で参照されたレコードの次のレコードに書き込まれます。

なお、書き出すデータはあらかじめFIELD文、LSET文、RSET文により準備しておかねばなりません。

バッファの最後を越えてリードまたはライトしようとすると「Field overflow」エラーとなります。

PUT KANJI

2

機　能

グラフィック画面に漢字を表示します。

書　式

PUT KANJI[(X,Y)],<漢字コード>[,<カラーコード>[,<論理演算子>[,<モード>]]]

文 例

```
10 SCREEN 5:X = 0:Y = 0:I =&H2121
20 PUT KANJI(X,Y),I
30 X = X+16:IF X = 256 THEN X = 0 ELSE 50
40 Y = Y+16:IF Y = 208 THEN A$= INPUT$(1):CLS:Y = 0
50 I = I+1
60 IF (I AND 255 OR 128)= 128 THEN I = I+160
70 IF I =&H2820 THEN I =&H3020
80 IF I<&H4F53 THEN 20
90 END
```

解 説

＜漢字コード＞はJIS漢字コードで、MSX2では&H2121から&H4F53、MSX2＋では&H2121から&H737Cの値を取ります。＜論理演算子＞についてはLINEを参照してください。＜モード＞は0〜2で次のような意味を持ちます。

0	16×16ノーマル（省略値）
1	偶数ラインだけを縦8ドットで表示
2	奇数ラインだけを縦8ドットで表示

モード1と2はインタレースモードで使います。漢字ROMがシステムにない場合は何も表示しません。PUT KANJIはSCREEN 5以降でのみ有効です。
この例では、漢字ROMに入っている文字を順に画面に表示します。途中、余分な空白部分は飛ばしています。

参 照

LINE、SCREEN

PUT SPRITE

機 能

スプライトパターンを画面に表示します。

書 式

PUT SPRITE＜スプライト面番号＞[,[STEP](X,Y)][,＜カラーコード＞][,＜スプライトパターン番号＞]

文 例

```
10 SCREEN 1,2
20 SPRITE$(0)=STRING$(32,CHR$(&B11111111))
30 PUT SPRITE0,(100,100),1,0
40 END
```

解 説

このプログラムはSPRITE$命令によってスプライトパターン番号0に定義されている四角い模様を、スプライト面0番を使って、画面の(100,100)の位置に、黒で表示します。なお、プログラムが終わってもスプライトは画面から消えません。スプライトを消すには、Y座標に209か208(SCREEN 4以降では217か216)を指定してPUT SPRITEを実行するか、SCREEN命令の第2パラメータで再度スプライトの大きさを指定します。スプライト面番号は0〜31の値で指定します。1つのスプライト面には、1つのパターンだけしか表示できません。すでにパターンが表示してある面を指定すると、先に表示されていたパターンは消されます。

スプライト面の0〜31は同時に表示できますが、横方向に5つ以上(SCREEN 4以降では9つ以上)のパターンが並ぶと、5つ目以降(SCREEN 4以降では9つ目以降)のパターンは画面から消えます。

(X,Y)で、パターンを表示する位置を指定します。Xは−32〜255の値を、Yは-32〜191の値(SCREEN 4以降では〜211)を指定します。STEPをつけて位置を指定することもできますが、これについてはLINEを参照してください。(X,Y)を省略すると、そのスプライト面で使われている値が使われます。この(X,Y)はスプライトパターンの左上隅の位置を指しています。

パターンが同じ位置に重なると、スプライト面番号の小さい方のパターンが手前に重なって表示されます。

Yに208(SCREEN 4以降では216)を指定すると、そのスプライト面以降のパターンが画面から消されます。Yに209(SCREEN 4以降では217)を指定すると、指定したスプライト面に表示されていたパターンが消されます。

色は、表示するパターンの色で0〜15の値で指定します。省略するとCOLOR文で指定している前景色となります。カラー番号と色との対応については、COLORを参照してください。またSCREEN 4以降の各スプライト面では、1ライン毎に色を付けることができます。これについては、COLOR SPRITEを参照してください。

スプライト番号は、SPRITE$で指定した番号です。

参 照

COLOR、COLOR SPRITE、SPRITE$、SCREEN

READ

機　能

DATA で用意した数値や文字のデータを読み込み、変数に代入します。

書　式

READ<変数>[, <変数>・・・]

文　例

```
10 READ A$,A,B$,B
20 PRINT B$,B,A$,A
30 DATA アカクラ,3.5
40 DATA サ゛ オウ,2.3
```

解　説

30 行、40 行のデータを READ で読み込み、「アカクラ」を A$、3.5 を A、「サ゛ オウ」を B$、2.3 を B にそれぞれ代入しています。

READ は、行番号の小さい DATA のデータから順に読み込み、指定した変数に代入していきます。DATA は、READ で読み出される数値や、文字を用意します。これらの数値や文字のことをデータといいます。データは、1 行では 255 文字以内で、カンマ（,）で区切って指定します。データを複数行指定することもできますが、その場合もそれらのデータは連続したものと見なされます。データは、整数、単精度実数、倍精度実数、文字定数のどれでもかまいませんが、READ 文で指定する変数の型とそれぞれ対応していなくてはなりません。データとして、文字列の先頭や最後に空白、ピリオド（.）、カンマ（,）があるときには、その文字列を引用符（"）で囲みます。

参　照

DATA、RESTORE

REM

機　能

プログラム中に注釈を入れます。

書　式

REM[<注釈文>]

文　例

REM MSX

解　説

REM に続く文字列は、プログラム中の注釈になるだけで、プログラムの実行には全く関係しません。プログラムを見やすくする覚え書きとして使用してください。REM の代わりにアポストロフィ（'）を用いることもできます。例えば、上の例を、

'MSX

としても、かまいません。

RENUM

機　能

プログラムの行番号を付け直します。

書　式

RENUM[<新行番号>][,<旧行番号>][,<増分>]

文　例

RENUM 200,100,20

解　説

旧行番号が 100 のところから行番号の付けかえが行われます。新しい行番号の最初の番号は 200 で、以後 20 ずつ増えた行番号が付けられます。新行番号の最初の番号（この場合、200）、新行番号の増加分（この場合、20）とも、省略すると 10 が採用されます。行番号の付けかえを始めるところ（この場合、100）を指定しないと、そのプログラムの先頭から新しい行番号に変わります。
例えば、

RENUM
プログラム全体の行番号を付けかえます。新しい行番号は 10 から始まり、20、30・・・と付けられます。

RENUM 100,50
行番号50に新しい行番号として100が付けられ、以降110、120・・・と付け直されます。

RENUM 100,,100
プログラム全体の行番号を付けかえます。新しい行番号は100から始まり、200、300・・・と付けられます。

RENUM は、GOTO、GOSUB、THEN、ELSE、ON～GOTO、ON～GOSUB、ERL などで使われている行番号も、新しい行番号に合わせて変更します。

RESTORE

機　能

READ で読む DATA を指定します。

書　式

RESTORE[<行番号>]

文　例

```
10 RESTORE 50
20 READ A$,A
30 PRINT A$,A
40 DATA アカクラ,3.5
50 DATA ザ゛オウ,2.3
```

解　説

RESTORE で行番号 (50) を指定し、READ でどの DATA から読み始めるかを決めます。これにより、A$には「ザオウ」、A には2.3が代入されます。
行番号が省略されると、READ はプログラムの最初の DATA から読み始めます。

参　照

READ、DATA

RESUME

機　能

エラー処理を終了し、元のプログラムの実行を再開します。

書　式

RESUME[0]または NEXT または<行番号>

文　例

```
RESUME 100
```

解　説

ON ERROR GOTO でジャンプしたエラー処理の操作を終了し、元のプログラムの 100 行から再び実行を再開します。RESUME 0 と指定すると、エラーが発生した行からプログラムが再開されます。この場合、0 は省略できます。RESUME NEXT と指定すると、エラーが発生した次の行からプログラムが再開されます。

参　照

ON ERROR GOTO

RETURN

機　能

サブルーチンから戻ります。

書　式

RETURN[<行番号>]

文　例

```
10 GOSUB 40
20 GOGUB 60
30 END
40 PRINT "40 ｷﾞｮｳ"
50 RETURN
60 PRINT "60 ｷﾞｮｳ"
```

```
70 RETURN 30
```

解 説

サブルーチンを終了し、そのサブルーチンを呼び出した次の行に戻ります。また、上の70行のように、RETURNの後ろに行番号を指定して、復帰する行を指定できます。上のプログラムが実行される順番は、10、40、50、20、60、70、30行となります。

なお、サブルーチンの中でCLEAR命令やMAXFILES命令等を使うと、RETURNによる復帰ができなくなりますので注意してください。

参 照

GOSUB、ON GOSUB

RIGHT$

機 能

文字列の右側から指定された数の文字列を取り出します。

書 式

RIGHT$(<文字列>, <数式>)

文 例

```
10 A$="SUPER HOME COMPUTER"
20 PRINT RIGHT(A$,8)
30 END
```

表示 COMPUTER

解 説

カッコ内の文字列(A$)の右側から、数値(8)で指定された数の文字列を取り出します。
指定した数が文字列の総文字数より大きいときは、文字列がそのまま取り出されます。

参 照

LEFT$、MID$

RND

機　能

<数式>によって指定された0以上1未満の乱数を発生します。

書　式

RND(<数式>)

文　例

PRINT RND(1)

[表示] .748299301545582

解　説

<数式>が正のときは、同じ乱数の並びの新しい乱数を返します。
<数式>が0のときは、同じ乱数の並びの前回使った値を返します。
<数式>が負のときは、その値に対応した固有の乱数の並びを作ります。
RNDの乱数の並びは、プログラムが実行されるたびに同じものが使われますが、RND(-TIME)を1度先頭に用いれば、毎回別の乱数の並びを作ることができます。

RUN

機　能

メモリにあるプログラムの実行を開始します。あるいはフロッピーディスクからファイルをメモリにロードし、そのプログラムを実行します。

書　式

RUN [<行番号>]またはRUN "<ファイルスペック>"[,R]

文　例

RUN "NEWFILE"

解　説

<行番号>を指定すると、その行から実行が始まります。指定のない場合には、最も若い行番号から実行が始まります。
<ファイルスペック>が指定されると、指定されたプログラムをロード後、実行を開始

します。<ファイルスペック>中でデバイス名を省略すると、カレントドライブからロードされます。RUNコマンドを実行すると、すべての開いているファイルは閉じ、目的のプログラムをロードする前にメモリの内容がクリアされます。Rオプションを付けた場合には、すべてのデータファイルは開いたままになります。

SAVE

機　能

メモリのBASICプログラムを保存（セーブ）します。

書　式

SAVE <ファイルスペック>[,A]

文　例

```
SAVE "A:COM2",A
```

解　説

<ファイルスペック>で指定されるファイルにプログラムを書き込みます。デバイス名を省略すると、現在使用しているディスクドライブ（カレントドライブ）に出力されます。フロッピーディスクに保存する場合は、ファイル名は8文字以下で、拡張子は3文字以内で指定します。もし指定したファイル名がすでにディスク上に存在する場合は、古いファイルは消去され、新しいものに変更されます。
Aオプションが指定された場合には、プログラムはアスキー形式で保存され、オプションの指定がない場合には、内部表現形式（バイナリ形式）に圧縮されてプログラムの保存が行われます。アスキー形式の保存は、内部表現形式よりも、多くのディスクスペースを必要としますが、保存されたプログラムのファイルにロード以外の操作を加える場合には、アスキー形式で保存されていることが必要です。例えば、MERGEコマンドは、アスキー形式のファイルでなければ実行できません。また、アスキー形式で保存されたファイルは、データファイルとして読み出すこともできます。
カセットファイルに対しての保存では、内部表現(バイナリ)形式ではCSAVE、アスキー形式ではSAVEと使い分けますが、ディスクに対してはSAVE"　"で内部表現形式、SAVE" ",Aでアスキー形式というように、オプションによって使い分けます。

参　照

LOAD、CSAVE、MERGE

SCREEN

機 能

画面やスプライトの大きさなどについての設定を行います。

書 式

SCREEN[<画面モード>][,<スプライトサイズ>][,<キークリックスイッチ>][,<カセットボーレート>][,<プリンタオプション>][,<インタレースモード>]

文 例

SCREEN 1,2

解 説

使用する画面を32×24文字のテキストモードに設定し、スプライト模様の大きさを16×16に設定します。最初の値（1）が画面モードを、カンマに続く次の値（2）がスプライトの大きさを設定しています。
設定を行うには、以下の項目で説明している値をSCREENに続けて指定します。

■画面モード

12の画面モードのうちのどれを使うかを指定します。画面の指定をしてSCREENを実行すると、今まで表示されていた画面が消えて、新しい設定となります。ただし、SCREEN 10以上のモードでは互いに切り替えても画面の初期化はしません。またSCREEN 9は日本語バージョンでは使えません。BASICをスタートしたときには、32×24文字の画面が自動的に選ばれています。

モード	意 味
0	40×24（または80×26）文字のテキストモード
1	32×24文字のテキストモード
2	256×192点の高解像度グラフィックモード
3	64×48ブロックの低解像度グラフィックモード
4	256×192点の高解像度グラフィックモード
5	256×212点のビットマップグラフィックモード
6	512×212点のビットマップグラフィックモード
7	512×212点のビットマップグラフィックモード
8	256×212点のビットマップグラフィックモード
9	日本語バージョンにはない
10	256×212点のRGB、YJK混在モード、グラフィックルーチン処理あり
11	256×212点のRGB、YJK混在モード、グラフィックルーチン処理なし
12	256×212点のYJKモード

0、1のテキストモードではプログラムのリストを表示させたりできますが、線や絵を描くことはできません。2以降ではこれとは逆になります。0のテキストモードでは、スプライトを表示できません。また、2以降の画面を指定しても、プログラムが終了するとすぐにテキストモードに戻ってしまいます。2以降の画面を表示し続けたいときには、例えば、次のように指定します。

```
10 SCREEN 2
20 GOTO 20
```

このプログラムを実行すると、2の画面が表示され続けます。
それぞれのモードで使える色は、次のようになります。

0	16パレット中の2色
1	16パレット中の3色
2〜5、7	16パレット
6	4パレット
8	256色固定
10,11	0〜15のパレットおよびYJK
12	YJK

1つのパレットには、512色の中から1色を選んで割り当てることができます。パレットの割り当てについては、COLOR＝を参照してください。
MSX2＋ではスクリーン10、11、12が追加されています。10、11はRGB、YJK混在モードですが、10の場合はLINE文等のグラフィック命令を実行する場合に、RGB上の命令として処理し、アトリビュートビットのON／OFF等が考慮されます。11、12ではそのような処理はいっさいありません。YJK方式についてはVDPの表示モードに関する記述を参照してください。
また、スプライトはSCREEN 4以降では1ライン毎に色を付けることができ、水平線上に8つまで並べて表示することができます（COLOR SPRITE参照)。

■スプライトの大きさ

スプライトを画面に表示するときの大きさを設定します。また、SPRITE$でスプライトを作るときには、ここでの設定と同じ大きさにします（SPRITE$参照)。BASICをスタートさせた時点では、自動的に0が設定されています。

0	8×8の点でスプライトを作る
1	8×8の点のスプライトを縦横2倍にして表示する
2	16×16の点でスプライトを作る
3	16×16の点のスプライトを縦横2倍にして表示する

画面のモードを変えずに、スプライトの大きさだけを設定するには、例えば、

次のように指定します。このときスプライトの座標データ等は初期化されます。

```
SCREEN ,2
```

SCREEN では、ほかに3番目、4番目、5番目、6番目の値を指定することによって、以下のような設定を行うことができます。

表 2.12　SCREEN 文の 3～6 番目のパラメータ

	意　味	指定する値	備　考
3番目	キーを押したときに音を出すか出さないかを設定する	0 =音を出さない 1 =音を出す	BASIC スタート時には1に設定されている
4番目	プログラムをカセットにセーブする時のボーレートの設定	1 = 1200bps 2 = 2400bps	BASIC スタート時には1に設定されている
5番目	使っているプリンタが MSX 標準のものかどうかの設定	0 = MSX 標準 0 以外= MSX 標準以外	0 以外を指定すると、グラフィック文字は空白、ひらがなはカタカナに置き換えて印字される。
6番目	インタレースモードの設定	0 =通常 1 =インタレース 2 = Even / Odd 交互 3 = Even / Odd インタレース	MSX2 以降のみ可能。モード1は同じページの内容をインタレース表示。モード2は2つのページの内容を同一ライン上に交互に表示。モード3は2つのページの内容をインタレース表示。モード2と3はディスプレイページが奇数でなくてはならない。

参　照

COLOR、COLOR SPRITE、CSAVE、SPRITE$

SET ADJUST　2

機　能

画面の表示位置を上下左右に調整します。

書　式

SET ADJUST(<X 座標>, <Y 座標>)

文　例

```
SET ADJUST(4,4)
```

解　説

調整した値はバッテリーバックアップされたSRAMに記憶されます。X、Yはそれぞれ−8〜+7の値です。

SET BEEP

2

機　能

BEEP音を設定します。

書　式

SET BEEP<音色>, <大きさ>

文　例

```
SET BEEP 2,2
```

解　説

設定した値はバッテリーバックアップされたSRAMに記憶されます。<音色>、<大きさ>はそれぞれ0〜3の値です。

SET DATE

2

機　能

<日付>を現在の日付として設定します。

書　式

SET DATE <日付>[, A]

文　例

```
SET DATE "90/01/01"
```

解　説

設定した値はバッテリーバックアップされたSRAMに記憶されます。オプションのAを付けると、この日付にアラームが設定されます。<日付>は、「85/04/03」のように年月日をスラッシュで区切ります。月日が1桁の場合は04のように2桁とします。
アラームの設定日時になったときに、何が起こるかは各機種によって異なります。

SET PAGE 2

機 能

VRAM上でページングを行います。

書 式

SET PAGE <ディスプレイページ>, <アクティブページ>

文 例

```
SET PAGE 0,1
```

解 説

<ディスプレイページ>とは実際にモニタに表示されるページで、<アクティブページ>はデータが書き込まれるページです。ページを切り替えてもその内容は初期化されません。ページはSCREEN 5と6では0〜3、SCREEN 7以上では0〜1が使用できます。

SET PASSWORD 2

機 能

パスワードを設定します。

書 式

SET PASSWORD <パスワード>

文 例

```
SET PASSWORD "sesame"
```

解 説

電源ONのとき<パスワード>を入力しなければそのシステムが使えないようにします。<パスワード>は6文字以内です。なお設定したパスワードを忘れてしまったときなどは、[GRAPH]キーと[STOP]キーを押しながらリセットします。この設定はSRAMに記憶されますが、SET PROMPTやSET TITLEで設定するのと同じ場所を使っているのでこれらの設定とは共存できません。最後に設定したものだけが有効となります。設定したパスワードを無効にするときは、

```
SET TITLE" "
```

を実行します。詳しくはSET TITLEを参照して下さい。

SET PROMPT 2

機能

プロンプトを設定します。

書式

SET PROMPT <プロンプト>

文例

```
SET PROMPT "Ready"
```

解説

BASICのプロンプト「Ok」を好きなように変更します。<プロンプト>は6文字以内です。この設定はSRAMに記憶されますが、SET PASSWORDやSET TITLEで設定するのと同じ場所を使っているのでこれらの設定とは共存できません。最後に設定したものだけが有効となります。プロンプトの設定が無効になった場合は、プロンプトは「Ok」となります。

SET SCREEN 2

機能

現在のWIDTH、SCREENの各パラメータを初期値として登録します。

書式

SET SCREEN

文例

```
SET SCREEN
```

SET SCROLL

2+

機　能

ハードウェアスクロールを行います。

書　式

SET SCROLL [<X座標>][,<Y座標>][,<左端マスク>][,<2ページ連続>]

文　例

```
SET SCROLL 100,100,1,1
```

解　説

MSX2+専用命令で、V9958のハードスクロールをサポートします。<X座標は>0～511、<Y座標>は0～255です。<左端マスク>が1なら画面の左端8ドット分は表示せず、0なら表示します。<2ページ連続>が1なら連続する2ページ(例えば、0、1)を横につながった1ページとして処理し、0なら単独ページ内のスクロールだけとします。2ページ連続の表示を行うときは表示ページが奇数ページでなくてはなりません。
ハードスクロールはSCREEN 1以降、2ページ連続の表示はSCREEN 5以降でのみ可能です。

SET TIME

2

機　能

<時刻>を現在の時刻として設定します。

書　式

SET TIME <時刻>[,A]

文　例

```
SET TIME "14:50:30"
```

解　説

設定した値はバッテリーバックアップされたSRAMに記憶されます。オプションのAを付けると、この時刻にアラームが設定されます。<時刻>は、「11:20:03」のように時分秒をコロンで区切ります。時刻が1桁の場合は03のように2桁にします。
アラームの設定日時になったときに、何が起こるかは各機種によって異なります。

SET TITLE

2

機 能

初期画面のタイトルと色を指定します。

書 式

SET TITLE<タイトル>[,<色番号>]

文 例

SET TITLE "MSX2+!"

解 説

<タイトル>は6文字以内で、ちょうど6文字のときは何かキーを押すまでタイトルのままになります。<色番号>は1〜4です。

番号	色
1	青
2	緑
3	赤
4	オレンジ

この設定はSRAMに記憶されますが、SET PASSWORDやSET PROMPTで設定するのと同じ場所を使っているのでこれらの設定とは共存できません。最後に設定したものだけが有効となります。
PASSWORD、PROMPT、TITLEの機能を使わないときは、

```
SET TITLE" "
```

を実行して下さい。

SET VIDEO

2

機 能

スーパーインポーズのモードを設定します。

書 式

SET VIDEO <モード>[, <YM>[, <CB>]]

文 例

```
SET VIDEO 2
```

解 説

<モード>は

0	内部同期、コンピュータ画面
1	外部同期、コンピュータ画面
2	外部同期、スーパーインポーズ
3	外部同期、外部ビデオ画面

となっています。

<YM>はモニタの画面密度で、0がノーマル、1がハーフトーンです。<CB>はVDPのカラーバスのモードで、0が出力、1が入力モードとなります。

この命令は、スーパーインポーズ機能のない機種では無意味です。

SGN

機 能

<数式>の符号を調べ、正の数なら1、0なら0、負の数なら−1を返します。

書 式

SGN(<数式>)

文 例

```
PRINT SGN(20)
```

[表示] 1

SIN

機 能

<数式>の正弦（サイン）を計算します。

書 式

SIN(<数式>)

文 例

PRINT SIN(3.1415927/2)

表示 1

SOUND

機 能

PSGに直接指定して音を出します。

書 式

SOUND<レジスタ番号>,<データ>

文 例

```
10 'チャンネル A
20 SOUND 0,254
30 SOUND 1,0
40 SOUND 8,8
50 'チャンネル B
60 SOUND 2,250
70 SOUND 3,0
80 SOUND 9,16
90 'チャンネル C
100 SOUND 6,4
110 SOUND 10,4
120 'キョウツウ
130 SOUND 7,&B011100
140 SOUND 11,172
150 SOUND 12,13
160 SOUND 13,12
```

解 説

このプログラムは、チャンネルA、Bから音を出し、チャンネルCからはノイズを出すものです。チャンネルAとCの音量は一定ですが、チャンネルBでは波形によって音量が変わるように指定しています。音を止めるには、CTRLキーを押しながらSTOPキーを押すか、次のように打ち込んでそれぞれリターンキーを押します。

SOUND 8,0	チャンネルAの音が止まる。
SOUND 9,0	チャンネルBの音が止まる。
SOUND 10,0	チャンネルCの音が止まる。

SOUNDでは、A、B、Cの三つのチャンネルごとに細かな指定をして、各自あるいは同時に音を出すことができます。指定は次の形で行います。

SOUND レジスタ番号,データ

レジスタ（SOUND用の特別のメモリ）は0〜13まであり、番号に応じてそれぞれの機能を持っているので、SOUNDでは行わせたい機能に応じてレジスタを選択します。
データは、それぞれのレジスタの情報となるものです。データはレジスタごとに1つずつ指定します。
3つのチャンネルで使うレジスタを機能別に示します。

表2.13 SOUND命令のレジスタ番号

チャンネル	周波数	ノイズ周波数	チャンネル設定	音量	波形周期	波形形状
A	0 1	6	7	8	11 12	13
B	2 3	6	7	9	11 12	13
C	4 5	6	7	10	11 12	13

SOUNDで音を出すときに必ず指定しなくてはならないのは、チャンネル設定、音量、それに周波数またはノイズ周波数です。
次に、各レジスタの機能をまとめてみます。

表 2.14 SOUND 命令のレジスタの機能

レジスタ	機 能
0、1	チャンネル A の周波数を 2 つのレジスタを使って指定する
2、3	チャンネル B の周波数を 2 つのレジスタを使って指定する
4、5	チャンネル C の周波数を 2 つのレジスタを使って指定する
6	ノイズ周波数を指定する
7	どのチャンネルから音あるいはノイズを出すかを指定する
8、9、10	音量の指定と、波形を使うかどうかの指定を行う
11、12	波形の周期を 2 つのレジスタを使って指定する
13	波形の形状を指定する

■ **周波数の設定**

各チャンネルの周波数 f_T に対応するレジスタの値 (T_P) は、次のようにして求めます。

$T_P = f_{clock} / 16 \times f_T$

f_{clock}　1.78977MHz (1.78977×10^6)

f_T　周波数

例えば、440Hz の周波数を出したいときは、先の式の f_T に 440 をあてはめます。

$T_P = 1.78977 \times 10^6 / 16 \times 440$

T_P = 約 254

この T_P の値を 256 で割った答がレジスタ 1、3、5 の値、余りがレジスタ 0、2、4 の値になります。

254/256 ・・・答 (この場合、0)　レジスタ 1、3、5 への値
　　　　・・・余り(この場合、254)　レジスタ 0、2、4 への値

余りを求めるには BASIC の MOD を使って次のように指定すると簡単です。

```
PRINT 254 MOD 256
```

また、TP の値が 32767 を越える場合には、次のように指定します (例えば、32768)。

```
PRINT 32768-INT(32768/256)*256
```

チャンネル A にこれらの周波数を指定するときは、次のようにします。

```
SOUND 0,254
```

```
SOUND 1,0
```

データが0のときにも、前に指定した値が残っていることがあるので、必ず指定を行うようにします。

■ノイズ周波数の設定

ノイズ周波数 f_N に対応するレジスタの値（N_P）は、次のようにして求めます。

$N_P = f_{clock} / 16 \times f_N$

f_{clock}　1.78977MHz（1.78977×10^6）

f_N　ノイズ周波数（Hz）

N_P　レジスタ6に指定する値

30KHz のノイズを出力するには、上の式の f_T に 30 をあてはめて、

$N_P = 1.78977 \times 10^6 / 16 \times (30 \times 10^3)$

$N_P =$ 約4

レジスタ6へ入れる値は、4となります。SOUND で指定するには次のように行います。

```
SOUND 6,4
```

レジスタ6は、三つのチャンネルで共通です。

■チャンネルの設定

どのチャンネルから音あるいはノイズを出すのかを指定します。指定はチャンネルに対応するビットを0にします。

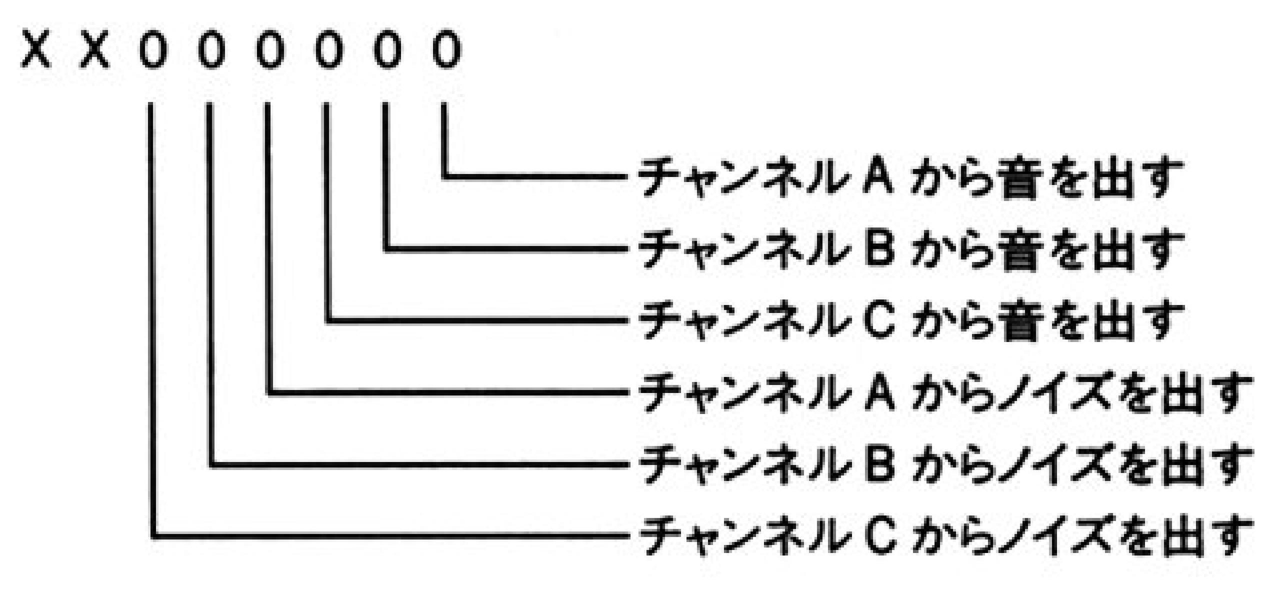


図2.16　SOUND 文のチャンネル設定

例えば、チャンネルAから音とノイズを出し、チャンネルBから音を出すという指定をするときの値は&B110100となります。
SOUND で指定すると、次のようになります。

SOUND 7,&B110100

レジスタ7のチャンネル指定の場合、このように2進数で指定した方がわかりやすく簡単です。

■ **音量設定**

音量の設定では、音の大きさと音量を波形に応じて変化させるかどうかの指定を行います。

音量は、0〜15の値で指定します。15のとき最大音量となり、0のときには、音が出ません。

音量を波形で変化させるときには16を指定します。

チャンネルBで波形変化を行うには、次のように行います。

SOUND 9,16

■ **波形周期の設定**

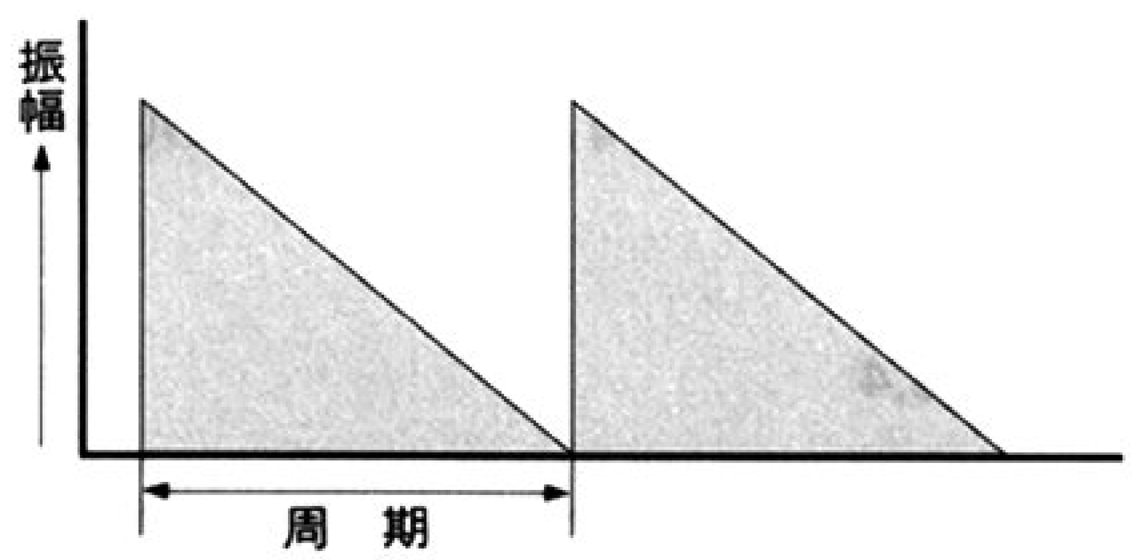


図 2.17 波形周期の設定

波形の周期EPは次のようにして求めます。

$E_P = f_{clock} \times T_E / 256$

f_{clock}	1.78977MHz (1.78977×10^6)
T_E	音が増加・減少する周期（秒）
E_P	周期

例えば、音の波が減少して0になるまでの時間を2秒にするとします。そのときに指定する値は次のようになります。

$E_P = 1.78977 \times 10^6 \times 2 / 256$

E_P = 約14000

この値を256で割ったときの答がレジスタ12への値、余りがレジスタ11への値です。

14000/256 ・・・答 （この場合、54） レジスタ12への値
・・・余り（この場合、176） レジスタ11への値

余りを求めるにはBASICのMODを使って次のように指定すると簡単です。

```
PRINT 14000 MOD 256
```

また、TPの値が32767を越える場合には、次のように指定します（例えば、32768）。

```
PRINT 32768-INT(32768/256)×256
```

この値をSOUNDで指定するには、次のように行います。

```
SOUND 11,176
SOUND 12,54
```

波形の周波数は三つのチャンネルで共通のレジスタに指定します。

■波形の形状の設定

波形の形には次の図のようなものがあります。

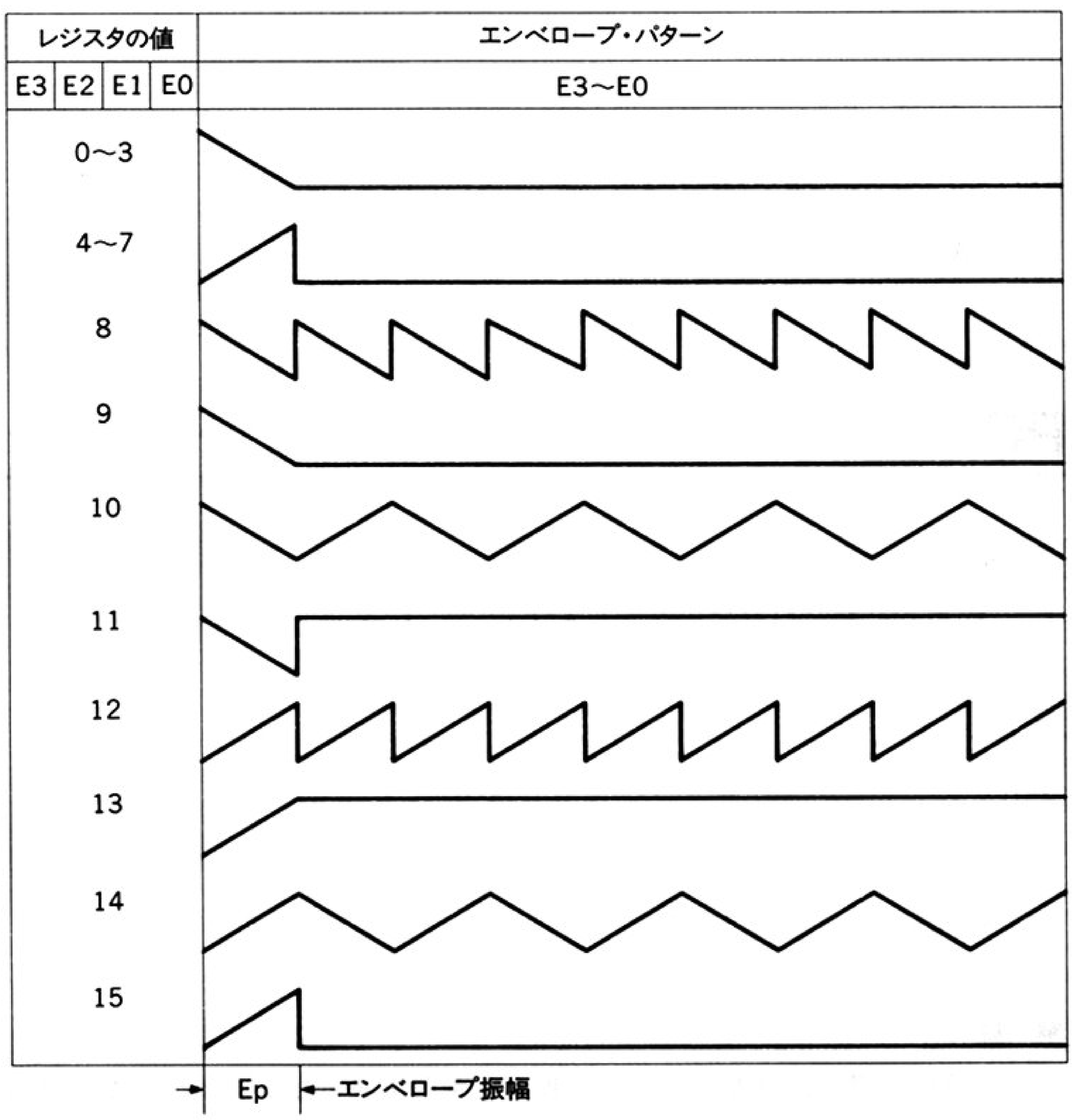


図2.18　波形形状の設定

この中から選んで値を指定します。値12の波形を指定するには、次のように行います。

```
SOUND 13,12
```

波形の形は三つのチャンネルで共通のレジスタに指定します。

SPACE$

機 能

指定した数の空白からなる文字列を作ります。

書 式

SPACE$(<数式>)

文 例

```
10 A$="MSX"
20 B$= SPACE$(5)
30 PRINT A$+B$+A$:END
```

表示 MSX　　MSX

SPC

機 能

指定した数だけ空白を表示します。

書 式

SPC(<数式>)

文 例

```
PRINT SPC(10);"TEST"
```

解 説

TABの場合は画面の左端から数えて空白を表示しますが、SPCは指定された位置から空白を表示します。SPCで指定できる値は、0から255です。

参照

SPACE$、TAB

SPRITE$

機能

スプライトで表示するパターンを定義します。

書式

SPRITE$(<スプライトパターン番号>)

文例1

```
10 SCREEN 1,0
20 FOR I = 1 TO 8
30 READ A$:B$= B$+CHR$(VAL("&B"+A$))
40 NEXT I
50 SPRITE$(0)= B$
60 PUT SPRITE0,(128,96),3,0
100 DATA 00010000
110 DATA 00111000
120 DATA 01111100
130 DATA 11111110
140 DATA 11111110
150 DATA 00010000
160 DATA 00010000
170 DATA 00010000
```

解説

SPRITE$の指定は次の形で行います。

SPRITE$(スプライト番号)=文字列データ

スプライト番号は、SCREEN でスプライトサイズを 0 か 1 に指定しているときには 0～255、スプライトサイズを 2 か 3 に指定しているときには 0～63 の値を付けます。
スプライトのパターンは、SCREEN のスプライトサイズの指定を 0 か 1 にしているときは 8×8 の点、2 か 3 にしているときには 16×16 の点で作ります。例えば、上の例では、10 行で SCREEN 1, 0 と指定しているので、スプライトは、8×8 の大きさになります。

スプライトのパターンを定義するときには、上の例のように0と1とで表したDATAを使うとわかりやすいものになります。1のところが画面に表示される部分で、0のところは表示されません。パターンの定義は、1バイトずつ行われます。

100行から170行は、1行が1バイト分になっているので、上から順番に文字変数に代入していき、8つとも代入されると、その文字変数の値はスプライトのパターンを表すものとなります。

上の例では、30行でデータ文からA$に値を読み込み、さらにA$の値が表すキャラクタコードの文字をB$に入れています。この行でA$を、"&B"+A$として2進数として扱うのは、100行から170行で指定しているデータが2進数に相当するものだからです。20行の指定でB$には100行から170行の8列のデータが入れられます。このB$の値がスプライトのパターンを表わすものです。50行で、スプライト番号0番としてB$の値を定義しています。60行のスプライトの表示の方法については、PUT SPRITEをみてください。

上の例の10行をSCREEN 1,1のように変えると、表示されるスプライトが倍の大きさになります。

文例2

```
10 SCREEN 1,3
20 FOR I = 1 TO 16
30 READ D$
40 A$= A$+CHR$(VAL("&B"+LEFT$(D$,8)))
50 B$= B$+CHR$(VAL("&B"+RIGHT$(D$,8)))
60 NEXT I
70 SPRITE$(0)= A$+B$
80 PUT SPRITE0,(128,96),8
100 DATA 0000000000011110
110 DATA 0000100000101001
120 DATA 0001011111101101
130 DATA 0001111111111001
140 DATA 0011111011111111
150 DATA 0001111111111000
160 DATA 0000001111111000
170 DATA 0000011111110000
180 DATA 0000001111100010
190 DATA 0000000111100100
200 DATA 1100001111100100
210 DATA 0011111111110010
220 DATA 0000001111110010
```

```
230 DATA 0000001111110010
240 DATA 0000000111111100
250 DATA 0000011111110000
```

このように、SCREENのスプライトサイズの指定を2か3とすると、16×16の点でスプライトのパターンを作ることができます。このとき注意しなくてはならないのが、データを代入していく順番です。

データは、まず左側の8つ(1バイト)が上から順番に、一番下まで定義されます。この時点で16個のデータ（1バイト単位で）が定義されたことになります。次に、今度は右がわの8つが上から順番に、一番下まで定義されます。これで合計32個のデータが定義され、16×16のパターンを定義したことになります。そのため文字変数は、左側用のもの（A$）と右側用（B$）のものと2つ用意されています。70行でこれらのデータを合わせて、1つのパターンをスプライトとして定義しています。

SCREEN 4以降のSPRITEでは、1ライン毎に色を付けることができます。これについてはCOLOR SPRITEを参照してください。

参照

COLOR SPRITE、PUT SPRITE

SPRITE ON / OFF / STOP

機能

スプライトのパターンが重なったときにジャンプするかどうかを決定します。

書式

SPRITE ON
SPRITE OFF
SPRITE STOP

文例

```
10 SCREEN 1,1
20 SPRITE$(0)=STRING$(8,CHR$(&B11111111))
30 ON SPRITE GOSUB 90
40 PUT SPRITE 0,(100,100),1,0
50 SPRITE ON
60 X=255
```

```
70 PUT SPRITE 1,(X,100),6,0
80 X = X-1:GOTO 70
90 REM SUB
100 X = 255:SPRITE OFF
110 RETURN
```

解 説

50行でSPRITE ONが指定してあるので、最初にパターンが重なったときには、30行の指定によって90行へジャンプします。100行でSPRITE OFFが指定してあるので、以後ジャンプは行われません。このプログラムを終了するには、CTRL キーを押しながら STOP キーを押します（スプライトパターンを消すには、画面モードかスプライトモードを指定したSCREEN命令を実行してください）。

SPRITE　ONを指定すると、以後スプライトのパターンが重なったときには、ON SPRITE GOSUBで指定した行へジャンプします。

SPRITE OFFを指定すると、以後スプライトのパターンが重なっても、ジャンプは起こりません。

SPRITE STOPを指定すると、以後スプライトのパターンが重なったときのジャンプを一時中止し、SPRITE ONが指定されると直ちにジャンプします。

SCREEN 4以降のSPRITEで、COLOR SPRITEによって特殊なカラー番号を指定しておくと、その色の部分は割り込みを起こさないようにすることができます。このやり方については、COLOR SPRITEを参照してください。

参 照

COLOR SPRIRE、ON SPRITE GOSUB

SQR

機 能

<数式>の平方根（ルート）を返します。

書 式

SQR(<数式>)

文 例

```
PRINT SQR(2)
```

表示 1.414213562373

STICK

機 能

ジョイスティックがどの方向に押されているかを調べます。

書 式

STICK(<ジョイスティック番号>)

文 例

```
10 PRINT STICK(0):GOTO 10
```

解 説

STICK に続くカッコ内の数値には、0か1か2が入り、それぞれ次のように、何の方向を調べるかを指定します。

番号	意 味
0	カーソルキー
1	ポート1に接続されたジョイスティック
2	ポート2に接続されたジョイスティック

STICK 命令の実行によって得られる数値は、0から8までの9つで、それぞれ以下の図に示された方向と対応しています。

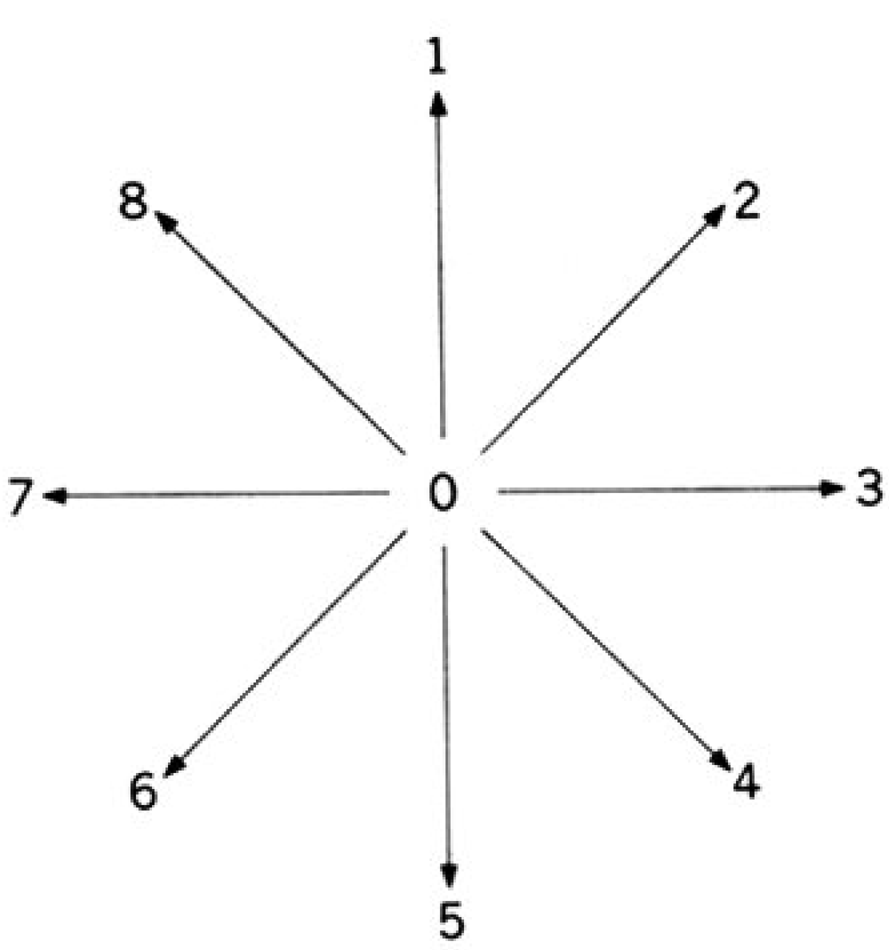


図 2.19　STICK 命令で得られる値

0は、ジョイスティックがどの方向にも向いていない（またはカーソルキーが何も押されていない）ことを示しています。

STOP

機　能

プログラムの実行を停止します。

書　式

STOP

文　例

```
10 PRINT "ハルカ゛ "
20 STOP
30 PRINT "キタ"
40 END
```

解　説

プログラムの実行を停止して、直接命令が入力できる状態にします。STOPはプログラムのどこにおいてもかまいません。STOP命令が実行されると、「Break in xxxx」(xxxxはSTOPされた行番号) と画面に表示されます。END命令と異なり、オープンされたファイルがあってもクローズはされません。プログラムを再開するにはCONT命令を実行します。

参　照

CONT

STOP ON / OFF / STOP

機　能

[CTRL] + [STOP] キーが押されたときにON STOP GOSUB文で指定された行に分岐するかどうかを決定します。

書　式

STOP ON
STOP OFF
STOP STOP

文 例

STOP ON

解 説

この命令は、ON STOP GOSUB によって定義された処理のサブルーチンの実行を許可するか（ON）、禁止するか（OFF）、一時停止するか（STOP）を決定します。

STOP ON を実行すると、[CTRL]＋[STOP] キーを押すごとに、ON STOP GOSUB により定義されている処理のサブルーチンが呼び出されます。

STOP OFF を実行すると、[CTRL]＋[STOP] キーを押しても、用意された処理のサブルーチンは実行されず、プログラムの実行が停止します。

STOP STOP を実行すると、[CTRL]＋[STOP] キーを押しても、用意された処理のサブルーチンは実行されませんが、再び STOP ON を実行することによって、処理が実行されます。

この命令は ON KEY(n) GOSUB 命令と KEY(n) ON/OFF/STOP 命令の関係と同じです。

STOP ON/STOP の状態では、プログラムによっては実行を停止できなくなるので注意して下さい。

参 照

ON STOP GOSUB、KEY(n) ON/OFF/STOP

STR$

機 能

数値を文字列に変換します。

書 式

STR$(<数式>)

文 例

```
10 A = 123:B =-45
20 C = A+B
30 C$= STR$(A)+STR$(B)
40 PRINT C,C$:END
```

[表示] 78　　　　123-45

解説

<数式>の数値を10進数の文字列に変換します。文字列に変換された「123」と「−45」は、もう数値ではないので、+によって加算されても「78」にはならず、「123−45」という文字列になります。

参照

VAL、STRING$

STRIG

機能

ジョイスティックのトリガボタンが押されたかどうか調べます。

書式

STRIG(<ジョイスティック番号>)

文例

```
10 PRINT "スペ゜ースキーヲオシテクダ゛サイ"
20 A = STRIG(0)
30 IF A THEN BEEP
40 GOTO 20
```

解説

スペースキーが押されると、−1が変数Aに代入されて30行が実行され、ビープ音が鳴ります。

STRIG命令は、指定したジョイスティックのトリガボタンまたはスペースキーが押されたかどうかを調べ、押されていれば−1を、押されていなければ0を返します。

STRIGに続くカッコ内には、0から4までの数値が入り、それぞれ以下に示すトリガボタンと対応しています。

番号	意味
0	トリガボタンのかわりにスペースキーを使う
1	ポート1に接続されたジョイスティックのトリガボタン1
2	ポート2に接続されたジョイスティックのトリガボタン1
3	ポート1に接続されたジョイスティックのトリガボタン2
4	ポート2に接続されたジョイスティックのトリガボタン2

ただし、ジョイスティックによってはトリガボタン2のないものもあります。

参 照

ON STRIG GOSUB、STRIG ON / OFF / STOP

STRIG(n) ON / OFF / STOP

機 能

ジョイスティックのトリガボタンが押されたときに、ON STRIG GOSUB 文によって指定された行に分岐するかどうかを決定します。

書 式

STRIG(<ジョイスティック番号>)ON
STRIG(<ジョイスティック番号>)OFF
STRIG(<ジョイスティック番号>)STOP

文 例

```
STRIG(0) ON
```

解 説

キーボードのスペースキーが押されると、ON STRING GOSUB によって指定された行番号以下の処理を実行します。この命令は、ON STRIG GOSUB によって定義された処理のサブルーチンの実行を許可するか (ON)、禁止するか (OFF)、一時停止するか (STOP) を決定します。
カッコ内の数値は、以下のようにジョイスティックのトリガボタンに対応しています。

番号	意 味
0	キーボードのスペースキー
1	ポート1に接続されているジョイスティックのトリガボタン1
2	ポート2に接続されているジョイスティックのトリガボタン1
3	ポート1に接続されているジョイスティックのトリガボタン2
4	ポート2に接続されているジョイスティックのトリガボタン2

ただし、ジョイスティックによっては、トリガボタン2のないものもあります。
STRIG ON を実行すると、指定されたトリガボタンが押されるごとに、ON STRIG GOSUB により定義されている処理のサブルーチンが呼び出されます。

STRIG OFF を実行すると、指定されたトリガボタンが押されても、用意された処理のサブルーチンは実行されません。
STRIG STOP を実行すると、指定されたトリガボタンが押されても、用意された処理のサブルーチンは実行されませんが、再び STRIG ON を実行することによって、処理が実行されます。

参照

ON STRIG GOSUB、KEY ON/OFF/STOP

STRING$

機能

<式2>で指定された文字だけでできた文字列を作ります。

書式

STRING$(<式1>, <式2>)

文例

```
10 A$= STRING$(10,"*")
20 PRINT A$:END
```

表示 **********

解説

<式2>で指定した文字を<式1>指定した数だけ連ねた文字列を作ります。<式2>の文字は 0〜255 までの数値でも構いません。この場合は、対応するキャラクタコードが使用されます。

SWAP

機能

2つの変数の値を交換します。

書式

SWAP<変数>, <変数>

文 例

```
10 A = 12:B = 34
20 PRINT "A =";A,"B =";B
30 SWAP A,B
40 PRINT "A =";A,"B =";B
50 END
```

表示 A = 12　　　B = 34
　　 A = 34　　　B = 12

解 説

行番号 30 の実行により、変数 A と変数 B の値が入れ替わります。変数の型は、文字型でも数値型でも構いませんが、2 つの変数の型は同じでなければいけません。

TAB

機 能

指定された桁まで空白を表示します。

書 式

TAB(<数式>)

文 例

```
PRINT TAB(10);"TEST"
```

解 説

指定した桁まで空白を出力します。指定した桁から表示を行いたいときに、PRINT や LPRINT などの命令の中で使います。指定できるのは、0 から 255 までの値です。
現在の出力位置がすでに指定された桁位置よりも大きいときは、何も出力しません。

参 照

SPC

TAN

機 能

＜数式＞の正接（タンジェント）を計算します。

書 式

TAN(＜数式＞)

文 例

```
PRINT TAN(3.14159/4)
```

表示 .99999867320603

TIME

機 能

時間を計ります。

書 式

TIME[=＜数式＞]

文 例

```
10 CLS:TIME=0
20 LOCATE 0,12:T%=TIME/60:PRINT T%
30 IF T%=60 THEN PRINT "1ﾌﾟﾝｹｲｶ" ELSE 20
40 END
```

解 説

TIMEは初期値を与えると、1/60秒ごとにその値を1つずつ増やしていきます。そのため、プログラム内の適当な場所にTIME=0を置いて、最後にTIMEの内容を調べれば、その間のプログラムの実行時間を計ることができます。TIMEに与える初期値は、0から65535までの値であれば、0でなくてもかまいません。また、フロッピーディスクを使用している間などはTIMEの値は更新されません。

上のプログラムでは、実行後の経過秒数が画面に表示されます。60秒経つと「1ﾌﾟﾝｹｲｶ」と表示されて、プログラムは終了します。

なお、上のプログラムの20行でT%としているのは、TIMEの値を60で割ったときの

余りを切り捨てるためです。

バッテリーバックアップされている時計の時刻や日付については、GET DATE、GET TIME、SET DATE、SET TIME を参照してください。

参 照

GET DATE、GET TIME、ON INTERVAL GOSUB、SET DATE、SET TIME

TRON、TROFF

機 能

プログラムの実行状態を追います。

書 式

TRON
TROFF

解 説

TRON を実行させてからプログラムを RUN させると、以後実行されたプログラムの行番号がカギカッコ([])付きで画面に表示されます。もとの状態に戻すときは、TROFF を用います。

USR

機 能

機械語で作られたプログラムを呼び出します。

書 式

USR[<番号>](<引数>)

文 例

```
I = USR3(J)
```

解 説

ユーザーの作成した機械語プログラムを呼び出します。機械語プログラムは、あらかじ

めメモリ中に準備されていなければならず、また DEF USR 命令により、その実行開始番地が指定されていなければなりません。
<番号>は DEF USR 命令により定義された番号に対応しており、0 から 9 が使えます。省略した場合は 0 とみなされます。カッコ内の引数は、BASIC から機械語プログラムへ値を引き渡すためのものです。

参照

DEF USR

VAL

機能

文字列を数値に変換します。

書式

VAL(<文字式>)

文例

```
10 A$="123":B$="-45"
20 C$= A$+B$
30 A = VAL(A$):B = VAL(B$)
40 C = A+B
50 PRINT C$,C:END
```

表示 123−45　　　　78

解説

カッコ内の文字列を数値に変換しています。A と B はそれぞれ「123」と「−45」が数値に変わったものなので、加算によって C は 78 という数になります。

参照

STR$

VARPTR

機　能

変数の格納されているメモリ番地、ファイルに割り当てられているファイルコントロールブロックの開始番地を求めます。

書　式

VARPTR (<変数名>)
VARPTR (#<ファイル番号>)

文 例 1

```
PRINT HEX$(VARPTR(A))
```

解 説 1

<変数名>で指定した変数のデータが格納されている変数領域のメモリ番地を求めます。このとき指定されている変数には値が代入されていなければなりません。

文 例 2

```
BVAD = VARPTR(# 1)
```

解 説 2

指定した<ファイル番号>に割り当てられているファイルコントロールブロックの開始番地を得ます。

参　照

メモリマップ

VDP

機　能

VDP (TMS9918、V9938、V9958) のレジスタへデータを書き込んだり、レジスタからデータを読み出したりします。

書　式

VDP(<レジスタ番号>)

文　例

VDP(1)=&HE1

解　説

VDP（ビデオディスプレイプロセッサ）のレジスタ1に、&HE1というデータを書き込みます。

読み書きできる範囲は次のとおりです。

	レジスタ0〜7(R/W)	ステータスレジスタ0(R)	レジスタ8〜(R/W)
MSX	0〜7	8	
MSX2	0〜7	8	9〜24(R8〜R23)
MSX2+	0〜7	8	9〜24、26〜28(R8〜R27)

参　照

BASE

VPEEK

機　能

VRAM上の指定した番地の内容を読み出します。

書　式

VPEEK(<番地>)

文　例

PRINT VPEEK(0)

解　説

読み出すデータを指定する番地は、SET PAGE命令で設定したページからのオフセットで、0〜&hFFFF(65535)の値です。返される値は0〜255です。詳しくはVRAMマップを参照して下さい。

参　照

SET PAGE、VPOKE

VPOKE

機　能

VRAM上の指定した番地にデータを書き込みます。

書　式

VPOKE＜番地＞,＜データ＞

文　例

```
VPOKE 0,65
```

解　説

データを書き込む番地は、SET PAGE命令で設定したページからのオフセットで、0～&HFFFF (65535) の値です。書き込む値は0～255です。詳しくはVRAMマップを参照して下さい。

参　照

SET PAGE、VPEEK

WAIT

機　能

入力ポートに指定した値が入るまで、プログラムの実行を中断します。

書　式

WAIT＜ポート番号＞,＜式1＞[,＜式2＞]

文　例

```
WAIT 1,&H22,&H22
```

解　説

＜ポート番号＞で指定した入力ポートから読み込んだデータと＜式2＞とのXORの結果を、＜式1＞でANDした結果が真になるまで、プログラムの実行を中断します。＜式2＞が省略されたときは、「0」が適用されます。ポート番号に関しては、I/Oマップを参照してください。

WAITは、ハードウェアと非常に密着した命令で、各々のMSXでは異なる動作をする場合があります。MSXの互換性を維持するため、広く公表するプログラムには使用しないでください。

WIDTH

機　能

1行に表示する文字数を設定します。

書　式

WIDTH<桁数>

文　例

WIDTH 30

解　説

テキストモードで画面に表示する、1行の文字数を30文字に設定します。40×24のモードでは1～40、32×24のモードでは1～32、80×26のモードでは1～80、までの値を指定できます。
文字数の設定は、各テキストモードでそれぞれ行います。例えば、32×24のテキストモードで、上の例のように30文字の指定を行っても、40×24のテキストモードでの設定は変わりません。
BASICをスタートした時点では、40×24のテキストモードで39文字、32×24のテキストモードで29文字が自動的に設定されます。

WIDTHステートメントは漢字モードの時には以下のように動作します。

漢字テキストモードの場合

■ **KANJI0かKANJI2の場合**
<桁数>の指定は26から64が有効です。

■ **KANJI1かKANJI3の場合**
<桁数>の指定は26から80が有効です。

上記以外の値が指定されると「Illegal function call」となります。

漢字グラフィックモードの場合

常に「Illegal function call」となります。

参照

CALL KANJI、SCREEN、SET SCREEN

3.14 エラーメッセージ一覧

メッセージ	エラーコード	説明
Bad drive name	62	ドライブ名の指定に誤りがある。
Bad FAT	60	ディスクのフォーマットが正常でない。
Bad file mode	61	OPENされたモードに対して、正常な入出力を行っていない。
Bad file name	56	ファイル名が適当でない。
Bad file number	52	オープンしていないファイル番号を指定した。指定したファイル番号がMAXFILES文で指定した数を越えている。
Bad sector number	63	セクタ番号に誤りがある。
Can't CONTINUE	17	CONTコマンドでプログラムの実行を再開できない。エラーが発生して終了したプログラムは再開できない。プログラムの一部を書き換えたり、CLEAR文などを実行すると、次に実行すべき箇所へのポインタが壊れて再開ができない。INPUT$関数を中断した場合再開できない。
Device I/O error	19	カセットレコーダ、プリンタなどへの入出力中にエラーが発生した。
Direct statement in file	57	アスキー形式のプログラムファイルをロードあるいはマージ中にプログラム以外のデータ（行番号がないもの）があった。
Disk full	66	ディスクの空き領域がない。
Disk I/O error	69	ディスクのデータ入出力中になんらかの障害が起きた。
Disk offline	70	ディスクが入っていない。
Disk write protected	68	ディスクが書き込み禁止状態なのに書き込もうとした。
Division by zero	11	0による除算を行おうとした。0に対して負のべき乗を行った。
FIELD overflow	50	FIELD文で定義したフィールドサイズの合計が256バイトを越えた。

メッセージ	エラーコード	説　明
File already exists	65	NAME文で指定したファイル名が、既にディスク上に存在している。
File already open	54	OPEN文で指定したファイル番号が、他のファイルで使用中である。
File not found	53	指定されたファイルが見つからない。
File not OPEN	59	OPEN文で開いていないファイルに対して入出力を行おうとした。
File still open	64	ファイルがまだクローズされていない。
Illegal direct	12	ダイレクトモードでは実行できないステートメント（実際にはMSX BASICにはない）をダイレクトモードで実行しようとした。
Illegal function call	5	ステートメントや関数の使い方が誤っている。引き数等が指定可能な範囲を越えている。
Input past end	55	ファイル中のデータを全て読み込んだ後、更に入力文を実行した。読み終わっているかどうかはEOF関数で調べる。
Internal error	51	BASIC内部でエラーが発生した。
Line buffer overflow	25	入力されたデータが許される文字数を越えた。
Missing operand	24	文中に必要なパラメータが欠けている。
NEXT without FOR	1	NEXT文に対応するFOR文がない。
No RESUME	21	エラー処理ルーチンにRESUME文がない。
Out of DATA	4	READ文で読み出せるデータがない。
Out of memory	7	プログラムが大き過ぎる、配列が大き過ぎるなどの原因でメモリが足りなくなった。
Out of string space	14	文字型変数用のメモリが足りなくなった。CLEAR文で対応する。
Overflow	6	演算結果や入力された数値が、型に応じた許容範囲を越えた。
Redimensioned array	10	配列変数を2重に定義しようとした。
Rename across disk	71	NAMEを異なるディスク間で行おうとした。
RESUME without error	22	エラー処理ルーチン以外でRESUME文を使用した。
RETURN without GOSUB	3	GOSUB文とRETURN文が正しく対応していない。
Sequential I/O only	58	シーケンシャルファイルに対してランダム入出力を行おうとした。
String formula too complex	16	指定した文字式が複雑過ぎる。式を分割する。
String too long	15	文字列が255文字より長い。

メッセージ	エラーコード	説　明
Subscript out of range	9	指定した配列変数の添字の値が、DIM 文で定義した配列変数の添字の最大値より大きい。
Syntax error	2	文の書き方を誤っている。カッコが対応していない、綴りが違う、記号が足りない等。
Too many files	67	ファイル数が 112 を越えた。
Type mismatch	13	数値型と文字型のデータを混用している。
Undefined line number	8	GOTO 文、GOSUB 文等で指定した行番号がない。削除しようとした行がない。
Undefined user function	18	DEF FN 文で定義していないユーザー定義関数を使用した。
Unprintable error	23,26～49,72～255	メッセージが定義されていないエラー。
Verify error	20	カセットにセーブされたプログラムがメモリ上のプログラムと内容が異なる。

4章 BASICの内部構造

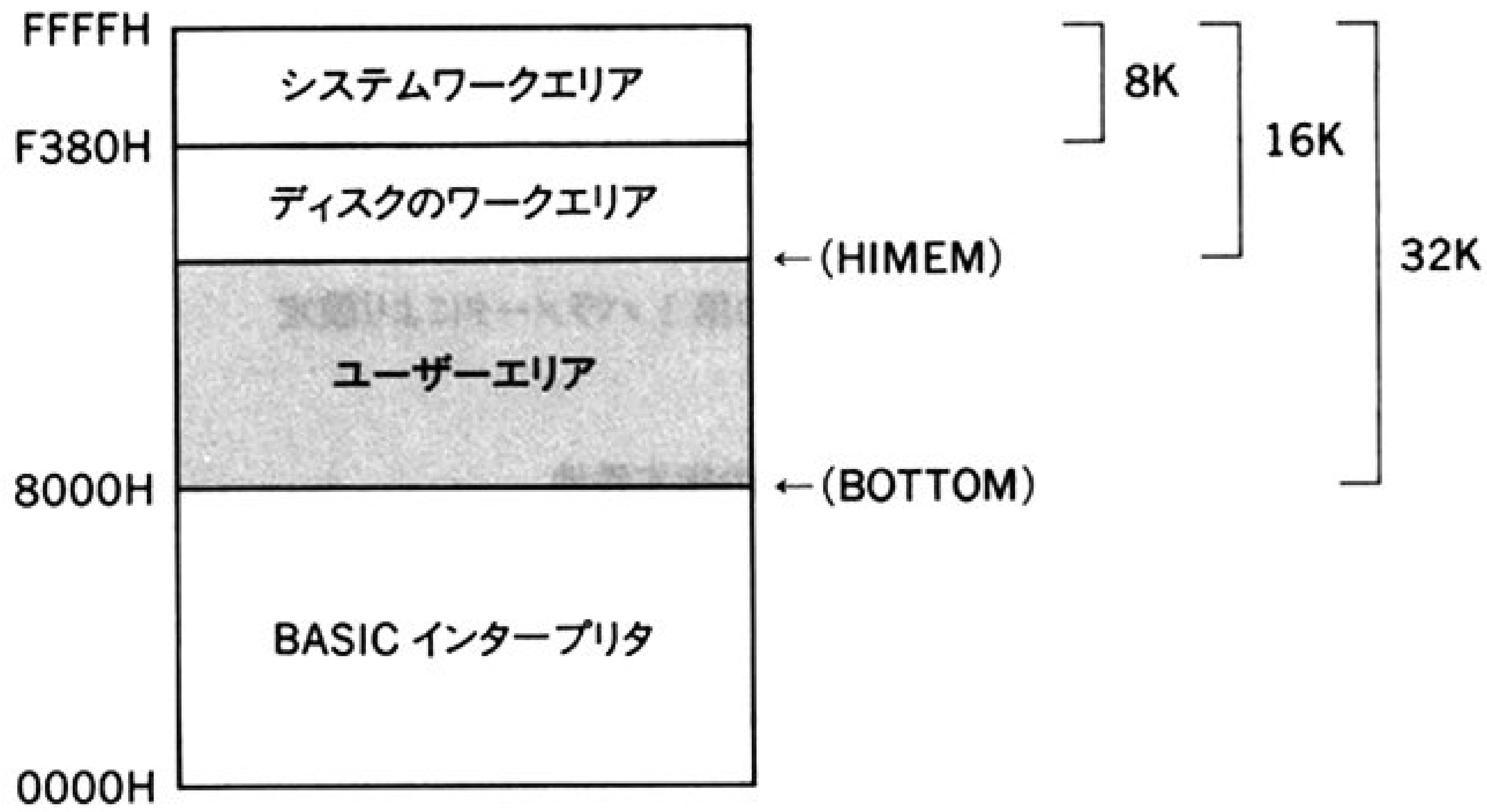
BASICをより高度に利用するためには、BASICインタープリタの内部構造に関する知識が不可欠です。この章では、BASICの内部構造について説明します。

4.1 ユーザーエリア

ユーザーエリアの下限アドレスは、MSX1ではRAM容量が8K、16K、32K、64Kといろいろだったため異なっていましたが、MSX2の場合、RAM容量は最低で64Kなので、ユーザーエリアの下限アドレスは常に8000H番地です。なお、これは【BOTTOM(0FC48H)】の内容から知ることができます。

ユーザーエリアの上限アドレスは、ディスクが実装されていない場合、F380H番地ですが、ディスクが実装されている場合（Disk BASIC使用時）はドライブ数やディスク容量などにより変わります。これは電源投入後、CLEAR文を実行する前の【HIMEM(0FC4AH)】の内容から知ることができます。

MSXを起動した時のメモリの状態を図2.20に示します。



注) RAM容量が32Kバイト以上であっても、BASICを使用している限りは32KバイトのRAMしか使用されない。ただし、MSX2においては、もう32KバイトをRAMディスクとしてBASICから使用可能となっている。

図2.20　BASICモードでのメモリの状態

MSXのアプリケーションプログラムを作成する場合、必ずHIMEMの内容を確かめ、最悪の場合でも暴走しないように対処してください。対処の方法としては、次のようなものが考えられます。

1. ワークエリアをリロケータブルにする
2. ワークエリアをBOTTOMから確保する
3. ドライブの数を減らすように指示して停止する

MSXはディスクが実装されている場合も、SHIFTキーを押しながらリセットすることにより、ディスクを切り離すことができます。また、ディスクが1ドライブだけ実装されている機種を起動すると、2ドライブのシミュレータ機能が働き、2ドライブ分のワークエリアが確保されます。このような時はCTRLキーを押しながらリセットすると、2ドライブシミュレータを無効にして、起動することができます。これにより、約1.5Kバイトのユーザーエリアを確保できます。

4.2 ユーザーエリアの詳細

BASIC使用時のユーザーエリアの状況を図2.21に、それらのエリアの開始番地へのポインタ情報を持つワークエリアを表2.15に示します。なお、このワークエリアは読み出し専用ですから、内容を書き換えた場合の動作に関しては保証されません。

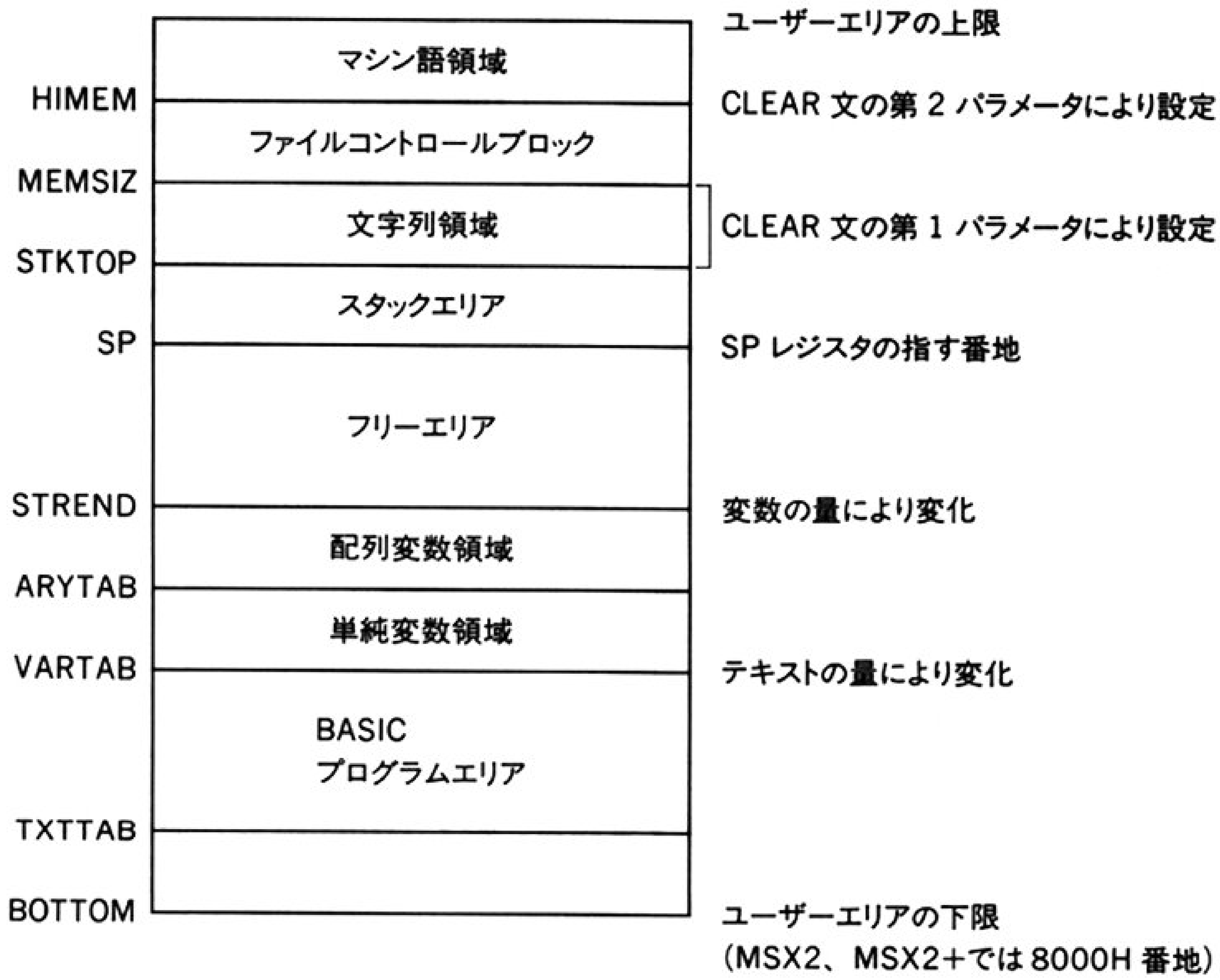


図 2.21　ユーザーエリアの状態

表 2.15　各エリアの開始と終了アドレスを持つワークエリア

エリア名	開始アドレス	終了アドレス
マシン語領域	【HIMEM(0FC4AH)】	ユーザーエリアの終わりまで
ファイルコントロールブロック	【MEMSIZ(0F672H)】	【HIMEM(0FC4AH)】−1
文字列領域 (文字列領域の空きエリアの先頭)	【STKTOP(0F674H)】 【FRETOP(0F69BH)】	【MEMSIZ(0F672H)】−1
スタックエリア	【SP レジスタ】	【STKTOP(0F674H)】−1
フリーエリア	【STREND(0F6C6H)】	【SP レジスタ】−1
配列変数エリア	【ARYTAB(0F6C4H)】	【STREND(0F6C6H)】−1
単純変数エリア	【VARTAB(0F6C2H)】	【ARYTAB(0F6C4H)】−1
プログラムエリア	【TXTTAB(0F676H)】	【VARTAB(0F6C2H)】−1
ユーザーエリア	【BOTTOM(0FC48H)】	リセット時の【HIMEM(0FC4AH)】−1

各ユーザーエリアの役割を以下に示します。

■ BASICのプログラムエリア

BASICで作成したプログラムは、ユーザーエリアの下限アドレス (MSX2では8000H番地) から格納され、その大きさはプログラムの量とともに変化します。

■変数領域

変数領域はBASICプログラムエリアのすぐ後ろに位置し、プログラム実行時に使用された変数の名前や値の保存のため確保されます。この時の変数の格納形式を図2.22（単純変数の場合）、図2.23(配列変数の場合)に示します。なお、配列変数をDIM文で宣言せずに使用すると添字が10の配列としてエリアが確保されます。ただし、4次元以上の配列はかならず宣言する必要があります。

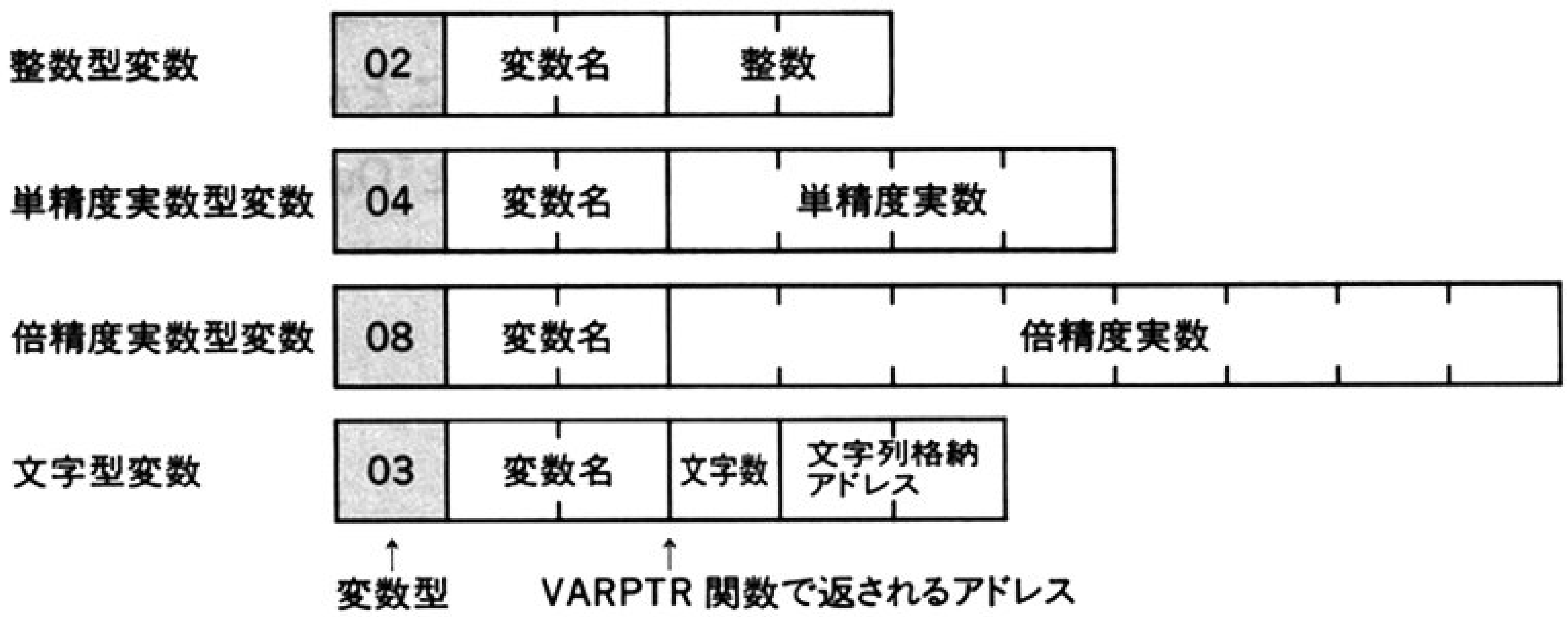
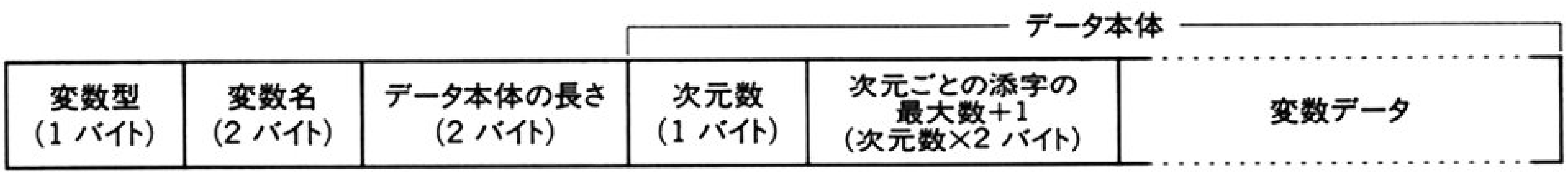
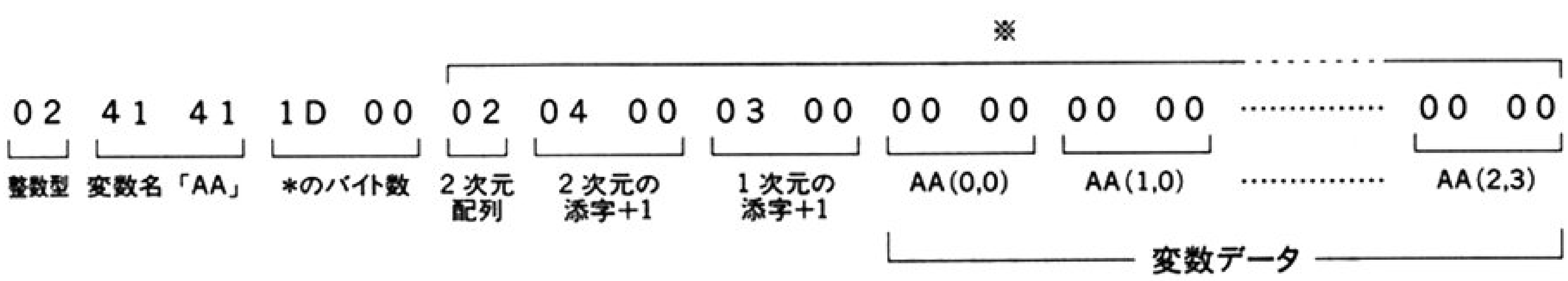


図2.22　単純変数の格納形式



例　DEFINT A:DIM AA(2,3)



注　意　変数データの形式は単純変数の格納形式と同じ。2バイトの数値は上位と下位バイトが逆に格納される。

図2.23　配列変数の格納形式

■フリーエリア

プログラムエリアや変数領域が大きくなったり、スタックが多くなって、フリーエリアがなくなると「Out of memory」エラーになります。なお、BASICのFRE関数でPRINT FRE(0)を

評価することにより、フリーエリアの大きさを知ることができます。

■スタックエリア

BASICが使用するスタックエリアです。GOSUB命令やFOR命令などの実行時に上位アドレス側から順に使用されます。

■文字列領域

文字列変数の内容を保存するために使用される領域で、上位アドレス側から順に使用されます。この領域の大きさはBASICのCLEAR文の第1パラメータで指定することができ、省略時は200バイトの大きさで確保されます。このエリアがすべて使われてしまうと「Out of string space」となります。BASICのFRE関数でPRINT FRE ("") を評価することにより、文字列領域の残りの大きさを知ることができます。

■ファイルコントロールブロック

ファイルの情報を保存する領域で、1ファイルごとに10BH (267) バイト分の領域が確保されます。ファイルいくつ分の領域を確保するかはBASICのMAXFILES文で指定でき、リセット時はファイル1つ分 (MAXFILES = 1) の領域が取られています。しかし、それとは別にSAVE、LOAD命令のために、常時かならず1つ分の領域が確保されているので、実際にはこれも含めて2つ分の領域が確保されていることになります。ファイルコントロールブロックの形式を表2.16に示します。

表2.16 ファイルコントロールブロック (FCB) の形式

変位	ラベル	意　味
+0	FL.MOD	ファイルがオープンされたモード
+1	FL.FCA	BDOS用FCBへのポインタ (Low)
+2	FL.LCA	BDOS用FCBへのポインタ (High)
+3	FL.LSA	バックアップキャラクタ
+4	FL.DSK	デバイスナンバー
+5	FL.SLB	インタープリタが内部で使用
+6	FL.BPS	FL.BUFの位置
+7	FL.FLG	いろいろな情報を持つフラグ
+8	FL.OPS	仮想的なヘッドの位置
+9～	FL.BUF	ファイルバッファ (256バイト)

■マシン語領域

マシン語プログラムを扱ったり、メモリを直接操作する場合の領域です。これらを行う場合は、CLEAR命令によって、あらかじめこの領域を確保しておく必要があります。

■ディスクのワークエリア

ディスク実装時に確保されるワークエリアを図2.24に示します。この領域はディスクが実装されていない場合は存在しませんので注意してください。なお、図中左のラベルは、そこにこのアドレスの情報があることを意味します。

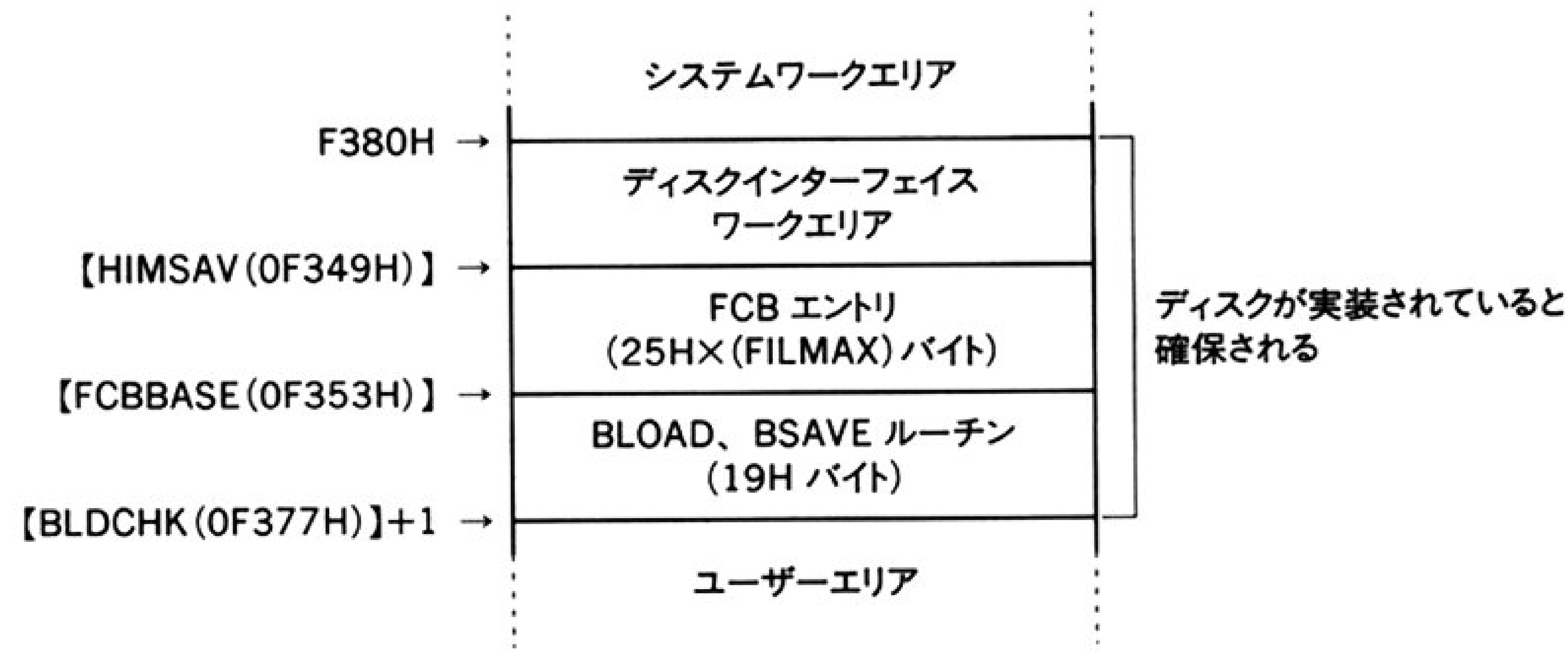
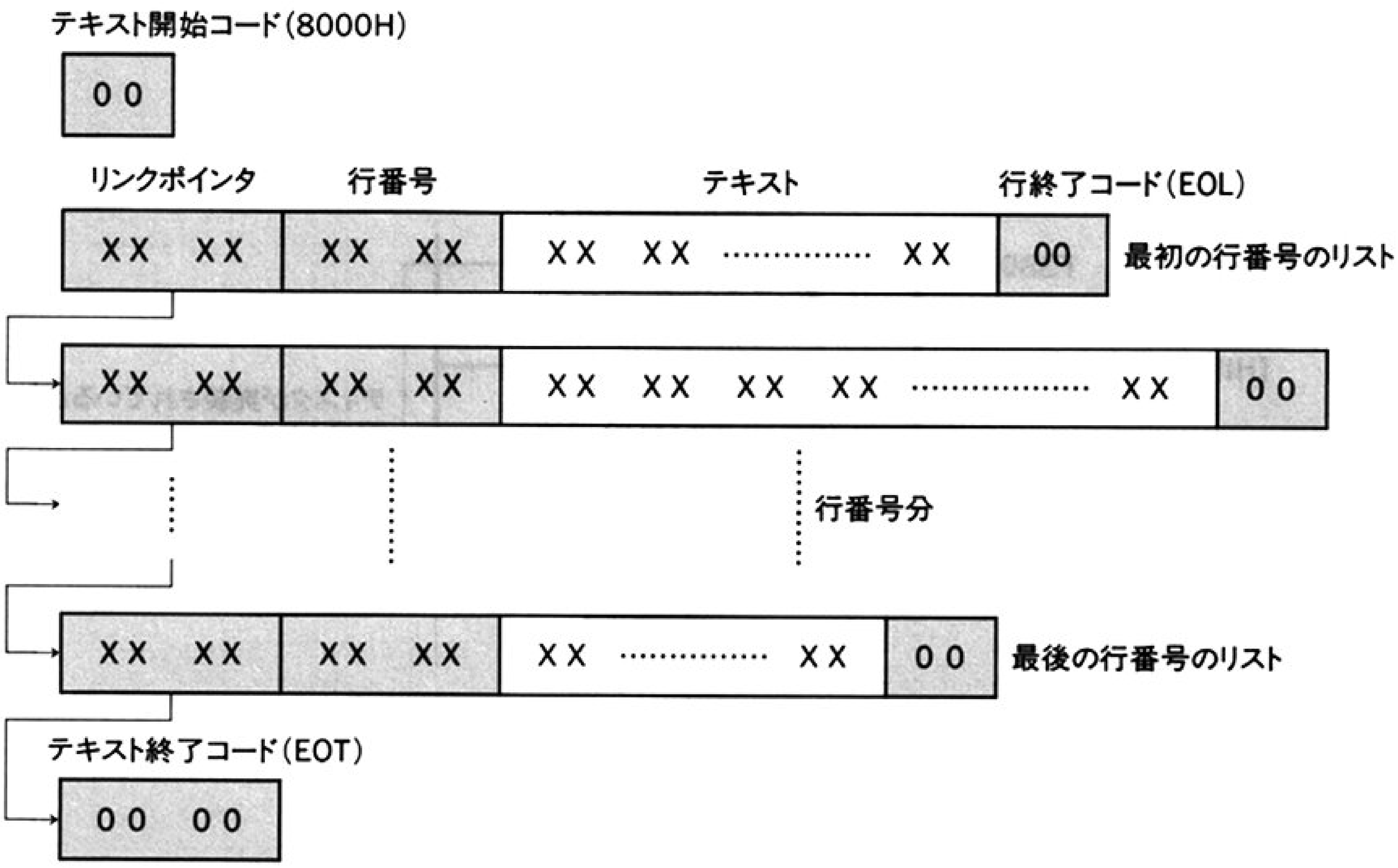


図2.24 ディスクのワークエリア

4.3 BASICプログラムの格納形式

BASICプログラムはメモリ上に図2.25のような形式で記憶されており、その内容には以下のような意味があります。



注 意　リンクポインタ、行番号は上位と下位バイトが逆に格納されています。

図2.25　テキスト格納形式

■リンクポインタ

次の行へのテキストポインタが絶対アドレスで書かれています。

■行番号

文字どおりプログラムの行番号が格納されており、通常は0～65529（0000H～FFF9H）の値が入ります。メモリを直接操作することにより65530以上の行番号を作ることも可能ですが、LISTコマンドはその行以降を表示しません。

■テキスト

ここにプログラム本体が中間コード形式で格納されています。中間コードに変換されるものは、

予約語（キーワード）、演算子、数値で、その他のもの（変数名や文字列定数など）はキャラクタコードの形で格納されます。中間コードを表2.17に、テキスト中の数値形式を図2.26に示します。

キャラクタコードについては、巻末の付録を参照してください。なお、グラフィックキャラクタは「CHR$(1)+(グラフィックキャラクタコード+64)」の2バイト（2文字分）で格納されますので、グラフィック文字を取り扱う時には注意が必要です。

表 2.17　中間コード一覧表

>	EE	CSNG	FF9F	GOTO	89	NAME	D3	SGN	FF84
=	EF	CSRLIN	E8	HEX$	FF9B	NEW	94	SIN	FF89
<	F0	CVD	FFAA	IF	8B	NEXT	83	SOUND	C4
+	F1	CVI	FFA8	IMP	FA	NOT	E0	SPACE$	FF99
-	F2	CVS	FFA9	INKEY$	EC	OCT$	FF9A	SPC (	DF
*	F3	DATA	84	INP	FF90	OFF	EB	SPRITE	C7
/	F4	DEF	97	INPUT	85	ON	95	SQR	FF87
^	F5	DEFDBL	AE	INSTR	E5	OPEN	B0	STEP	DC
¥	FC	DEFINT	AC	INT	FF85	OR	F7	STICK	FFA2
ABS	FF86	DEFSNG	AD	IPL	D5	OUT	9C	STOP	90
AND	F6	DEFSTR	AB	KEY	CC	PAD	FFA5	STR$	FF93
ASC	FF95	DELETE	A8	KILL	D4	PAINT	BF	STRIG	FFA3
ATN	FF8E	DIM	86	LEFT$	FF81	PDL	FFA4	STRING$	E3
ATTR$	E9	DRAW	BE	LEN	FF92	PEEK	FF97	SWAP	A4
AUTO	A9	DSKF	FFA6	LET	88	PLAY	C1	TAB (	BD
BASE	C9	DSKI$	EA	LFILES	BB	POINT	ED	TAN	FF8D
BEEP	C0	DSKO$	D1	LINE	AF	POKE	98	THEN	DA
BIN$	FF9D	ELSE	3AA1	LIST	93	POS	FF91	TIME	CB
BLOAD	CF	END	81	LLIST	9E	PRESET	C3	TO	D9
BSAVE	D0	EOF	FFAB	LOAD	B5	PRINT	91	TROFF	A3
CALL	CA	EQV	F9	LOC	FFAC	PSET	C2	TRON	A2
CDBL	FFA0	ERASE	A5	LOCATE	D8	PUT	B3	USING	E4
CHR$	FF96	ERL	E1	LOF	FFAD	READ	87	USR	DD
CINT	FF9E	ERR	E2	LOG	FF8A	REM	8F	VAL	FF94
CIRCLE	BC	ERROR	A6	LPOS	FF9C	RENUM	AA	VARPTR	E7
CLEAR	92	EXP	FF8B	LPRINT	9D	RESTORE	8C	VDP	C8
CLOAD	9B	FIELD	B1	LSET	B8	RESUME	A7	VPEEK	FF98
CLOSE	B4	FILES	B7	MAX	CD	RETURN	8E	VPOKE	C6
CLS	9F	FIX	FFA1	MERGE	B6	RIGHT$	FF82	WAIT	96
CMD	D7	FN	DE	MID$	FF83	RND	FF88	WIDTH	A0
COLOR	BD	FOR	82	MKD$	FFB0	RSET	B9	XOR	F8
CONT	99	FPOS	FFA7	MKI$	FFAE	RUN	8A	'	3A8FE6
COPY	D6	FRE	FF8F	MKS$	FFAF	SAVE	BA		
COS	FF8C	GET	B2	MOD	FB	SCREEN	C5		
CSAVE	9A	GOSUB	8D	MOTOR	CE	SET	D2		

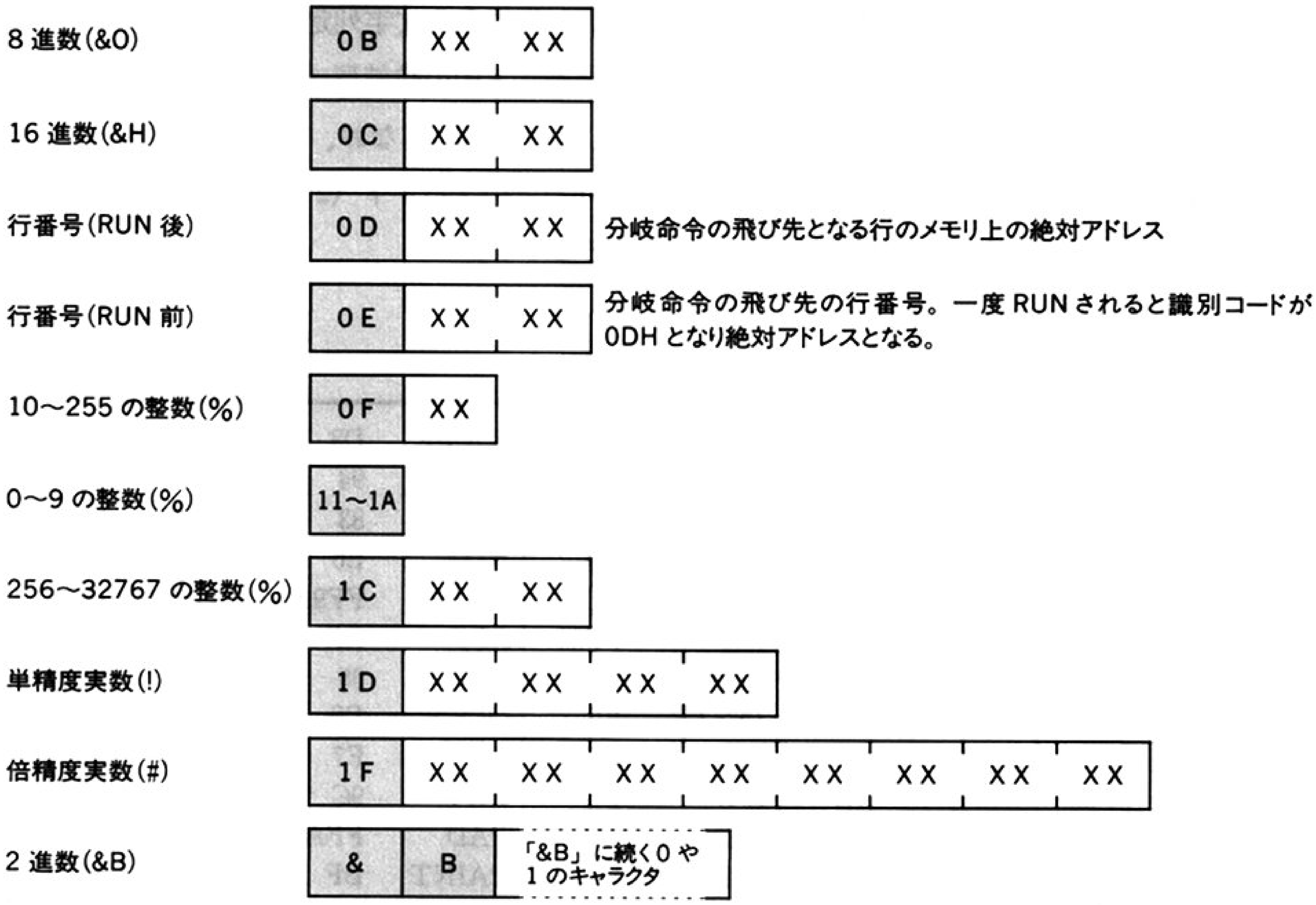


図 2.26　テキスト中の数値形式

数値は予約語や変数名との区別をつけるために「識別コード」と呼ばれる番号が割り付けられており、それを見ることで後に続く値がどのようなものかわかるようになっています。

2バイトの数値の場合は、上位と下位バイトが逆に格納されます。符号付き数値については、識別コードの前に＋または-の中間コードが置かれるのみであり、数値自体はかならず正の数で記憶されます。浮動小数点表記法についてはおおむね6.3のマスパック (Mathematical Package) の説明と同じですが、上記のように数値はかならず正の数で記憶される点に注意してください。2進数 (&B) については、識別コードは割り当てられておらず、アスキーコードそのままの形で記憶されます。

5章 マシン語とのリンク

これまで説明したように、MSX BASIC は強力な機能を備えていますが、さらに実行速度を上げたり、MSX2 のハードウェア機能を限界まで引き出す場合は、マシン語を使用しなければなりません。この章では、その必要が生じた時のために、BASIC からマシン語プログラムを呼び出す方法、およびその際に知っておくべき情報について説明します。

5.1 USR 関数

BASIC からのマシン語の呼び出しは、以下の手順で行います。USR 関数のカッコの中の値は、引数としてマシン語に渡すためのものです。引数は数式、文字式のいずれでもかまいません。

1. DEF USR 文で、マシン語プログラムの実行開始アドレスを指定する
2. USR 関数でマシン語を呼び出す
3. マシン語から BASIC に戻る場合は、RET 命令（C9H）を実行する

例

C000H 番地が実行開始アドレスであるマシン語プログラムを呼び出す。

```
DEF USR =&HC000
A = USR(0)
```

5.2 USR 関数の引数と戻り値によるデータの受け渡し

BASIC からマシン語に引数を渡す場合、マシン語側ではレジスタ A の内容でその型を判断します(表 2.18)。目的の値は引数の型により、図 2.27 のような形式で格納されていますから、それに応じて値を受け取ります。例として、文字列型の引数を受け取るプログラムをリスト 2.5 に示します。

表 2.18　レジスタ A に代入される引数の型

2	2 バイト整数型
3	文字列型
4	単精度実数型
8	倍精度実数型

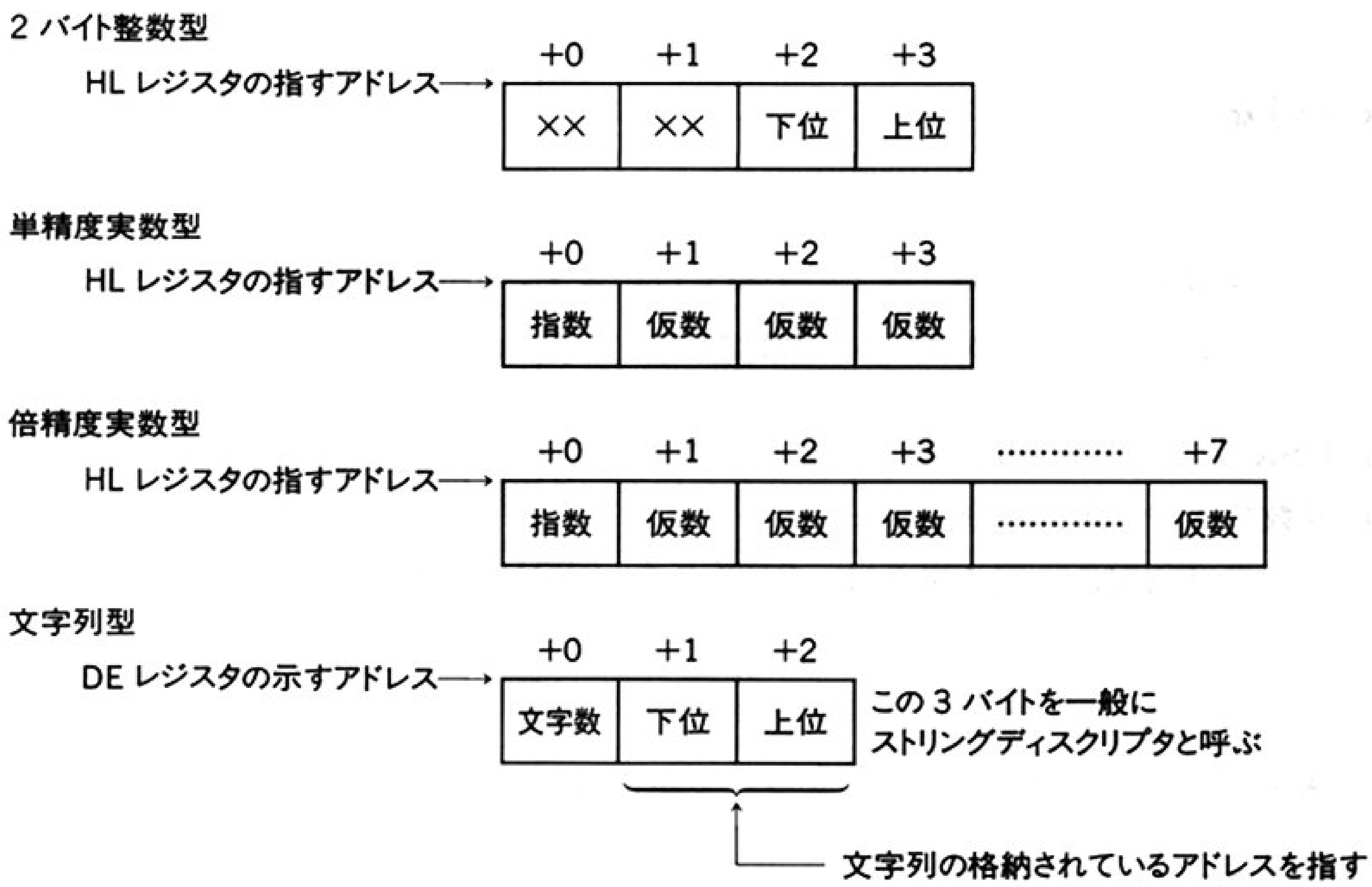


図 2.27　引数による値のわたされ方

リスト 2.5　文字型の引数の使用例

```
;
;       USR関数で渡した文字列をそのまま表示する
;        使用法：DEF USR =&Hxxxx：  A$= USR("STRING")
;
;
chput   equ     00a2h           ; character output

rdarg::
        cp      3
        ret     nz              ; parameter is not string

        push    de
        pop     ix              ; IX := string descriptor
        ld      a,(ix+0)        ; get string length
        ld      l,(ix+1)        ; get string pointer(low)
        ld      h,(ix+2)        ; get string pointer(high)
        or      a
        ret     z               ; if length = 0
rd1:
        push    af
        ld      a,(hl)          ; get a charcter
        call    chput           ; get a charcter
        pop     af
        dec     a
        ret     z
        inc     hl

        jr      rd1

        end
```

逆に、引数として渡された値をマシン語の中で処理することによって、USR関数の値としてBASICに引き渡すこともできます。この場合、【VALTYP(0F663H)】を変更することにより、戻り値をBASICからの引数の型以外のものに変更することも可能です。ただし、文字列型の場合、文字数を変えることはできません。

5.3 命令の増設

MSX では現在「CMD」と「IPL」という予約語が未使用となっており、この予約語のフック（それぞれ FE0DH、FE03H）を自分で作成した任意のマシン語ルーチンにジャンプするよう書き換えるだけで、新しい命令を作成することができます。簡単な例をリスト 2.6 に示します。

リスト 2.6　CMD 命令の増設

```
;
;        CMD コマンドの使用法(CAPS の ON/OFF)
;        コマンドの初期化：  DEF USR =&Hxxxx：  A = USR(0)
;        コマンドの使用法：  CMD
;
chgcap  equ     0132h           ; CAPS LAMP on/off
capst   equ     0fcabh          ; CAPS LOCK status
hcmd    equ     0fe0dh          ; CMD HOOK

; CMD コマンドの増設

start::
        ld      bc, 5           ; NEW HOOK SET
        ld      de,hcmd
        ld      hl,hdat
        ldir
        ret

; HOOK に書き込むデータ

hdat:
        pop     af
        jp      capkey
        nop

; CMD コマンドの実体

capkey:
        call    chgcap
        ld      a,(capst)
        cpl
        ld      (capst),a
        ret

        end
```

フックに書き込む最初の「pop af」は、「CMD」の実行時にスタックに積まれるエラー処理アドレスを捨てるためのものです。これを入れておかないと、「RET」命令でBASICに戻るはずが、エラー処理ルーチンにジャンプしてしまいます。ユーザールーチン内でエラーを出したい場合は、このアドレスを取っておくのも1つの方法です。

なお、これらのフックは本来は将来の拡張用にリザーブしてある場所なので、商用のアプリケーションプログラムで使用してはいけません。

5.4 BASICの内部ルーチン

より複雑なステートメントの拡張を行うには、CMD命令にも引数を渡せると便利です。マシン語を呼び出した際、HLレジスタがBASICテキスト中の「CMD」の次の位置を指していますから、これを利用して後に続く文字列の評価を行えば目的は果たせます。これらを行うために有効な内部ルーチンを以下で説明します。

CHRGTR（4666H / MAIN）

機 能

テキストから1文字取り出します。

コール手順

HL　　テキストを指すアドレス

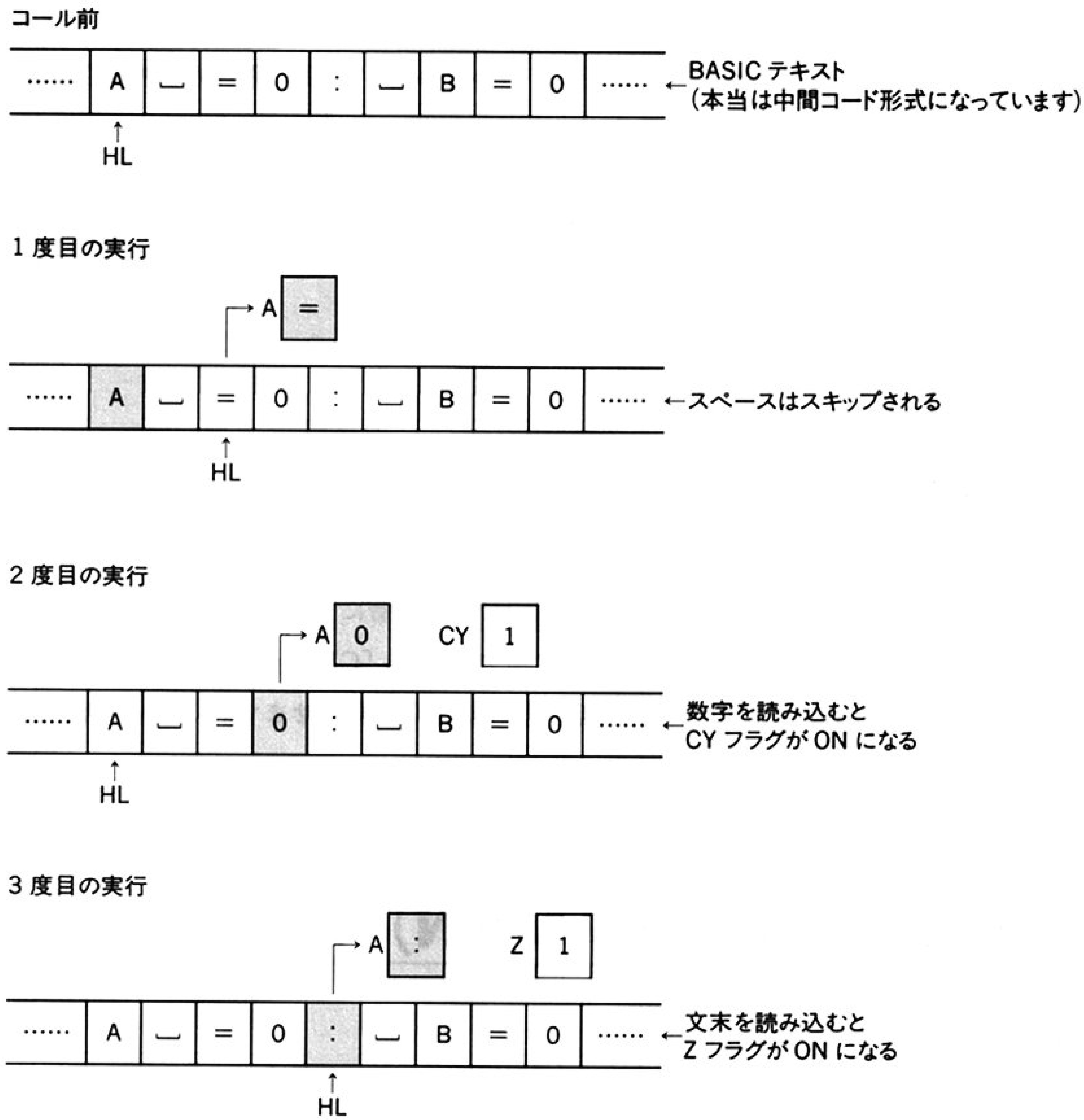
戻り値

HL　　取り出した文字のアドレス
A　　取り出した文字
Zフラグ　　文末（「：」または00H）ならばON
CYフラグ　　0〜9ならばON

変更レジスタ

解 説

(HL+1) の場所のテキストから1文字取り出します。スペースはスキップされます。



FRMEVL（4C64H / MAIN）

機　能

テキスト中の式を評価します。

コール手順

HL　　テキスト中の式の先頭アドレス

戻り値

HL　　式の次のアドレス

【VALTYP(0F663H)】　型に応じて2、3、4、8の値が入る

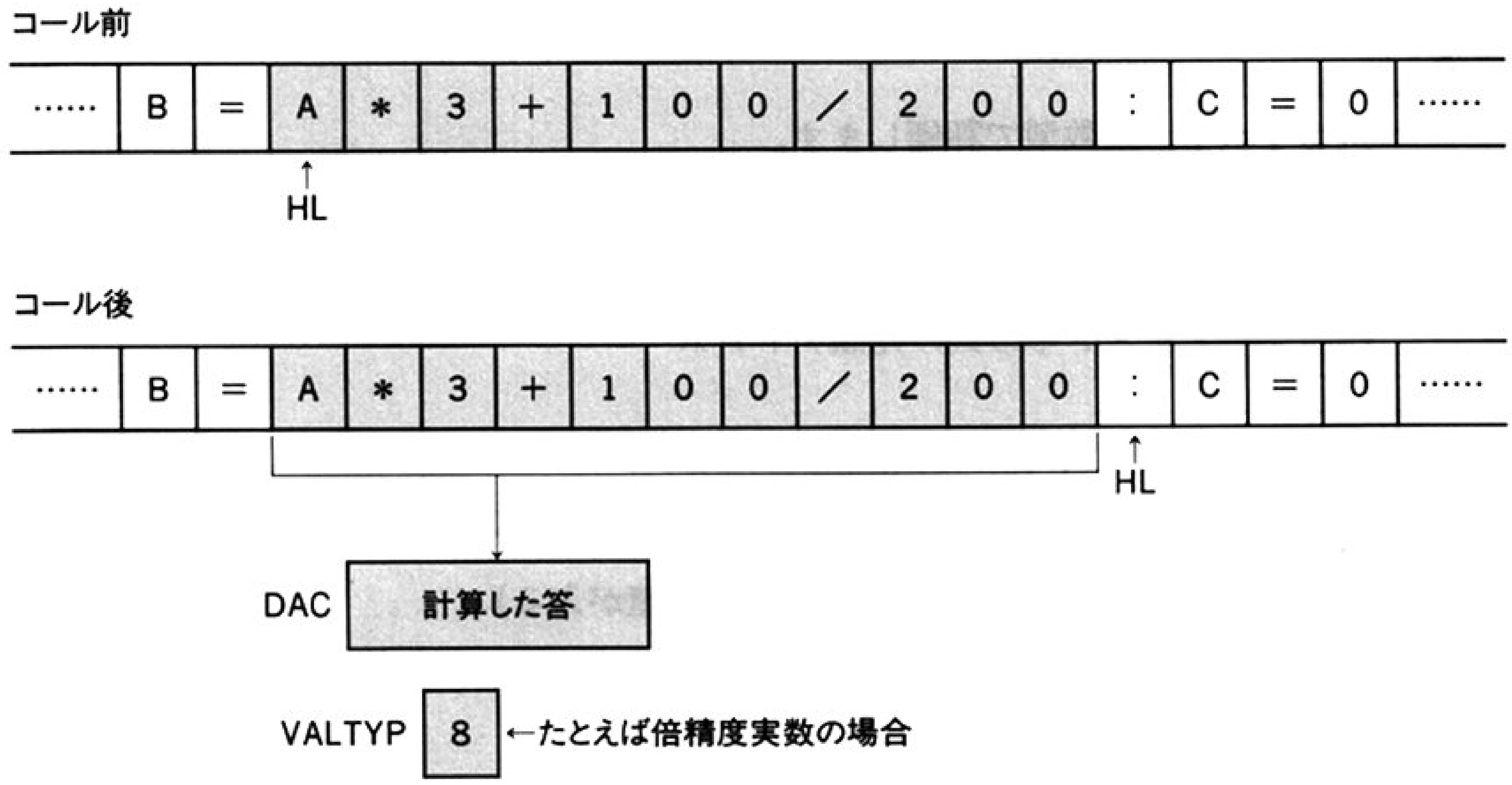
【DAC(0F7F6H)】　式の評価結果

変更レジスタ

すべて

解　説

式を評価し、型に応じて出力します「5.2 USR関数の引数と戻り値によるデータの受け渡し」参照。



FRMQNT（542FH / MAIN）

機　能

式を2バイト整数型で評価します。

コール手順

HL　　テキスト中の式の先頭アドレス

戻り値

HL　　式の次のアドレス

DE　　式の評価結果

変更レジスタ

すべて

解　説

式を評価し、整数型（INT）で出力します。結果が2バイト整数型の範囲に納まらない場合には、「Overflow」エラーを発生し、BASICコマンドレベルに戻ります。

GETBYT（521CH / MAIN）

機　能

式を1バイト整数型で評価します。

コール手順

HL　　テキスト中の式の先頭アドレス

戻り値

HL　　式の次のアドレス
A、E　　式の評価結果（AとEには同じ値が入る）

変更レジスタ

すべて

解　説

式を評価し、1バイト型で出力します。結果が1バイト整数型の範囲に収まらない場合には「Illegal function call」エラーを発生し、BASICコマンドレベルに戻ります。

FRESTR（67D0H / MAIN）

機　能

文字列を登録します。

コール手順

【VALTYP(0F663H)】　　型（文字型でないとエラーになる）
【DAC(0F7F6H)】　　ストリングディスクリプタへのポインタ

戻り値

HL　　ストリングディスクリプタへのポインタ

変更レジスタ

すべて

解　説

FRMEVL で得た文字列型の結果を文字列領域に登録し、そのストリングディスクリプタを得ます。文字列を評価した場合は、FRMEVL と組み合わせて以下のように使用するのが一般的です。

```
        :
call    frmevl
push    hl              ；テキストポインタをセーブする
call    FRESTR
ex      de,hl           ；ストリングディスクリプタを得る
pop     hl
ld      a,(de)          ；入力された文字列の長さを得る
        :
```

PTRGET（5EA4H / MAIN）

機　能

変数の格納アドレスを獲得します。

コール手順

HL　　テキスト中の変数名の先頭アドレス

【SUBFLG(0F6A5H)】　0=単純変数、0以外=配列変数

戻り値

HL　　変数名の次のアドレス

DE　　目的の変数の内容が格納されているアドレス

変更レジスタ

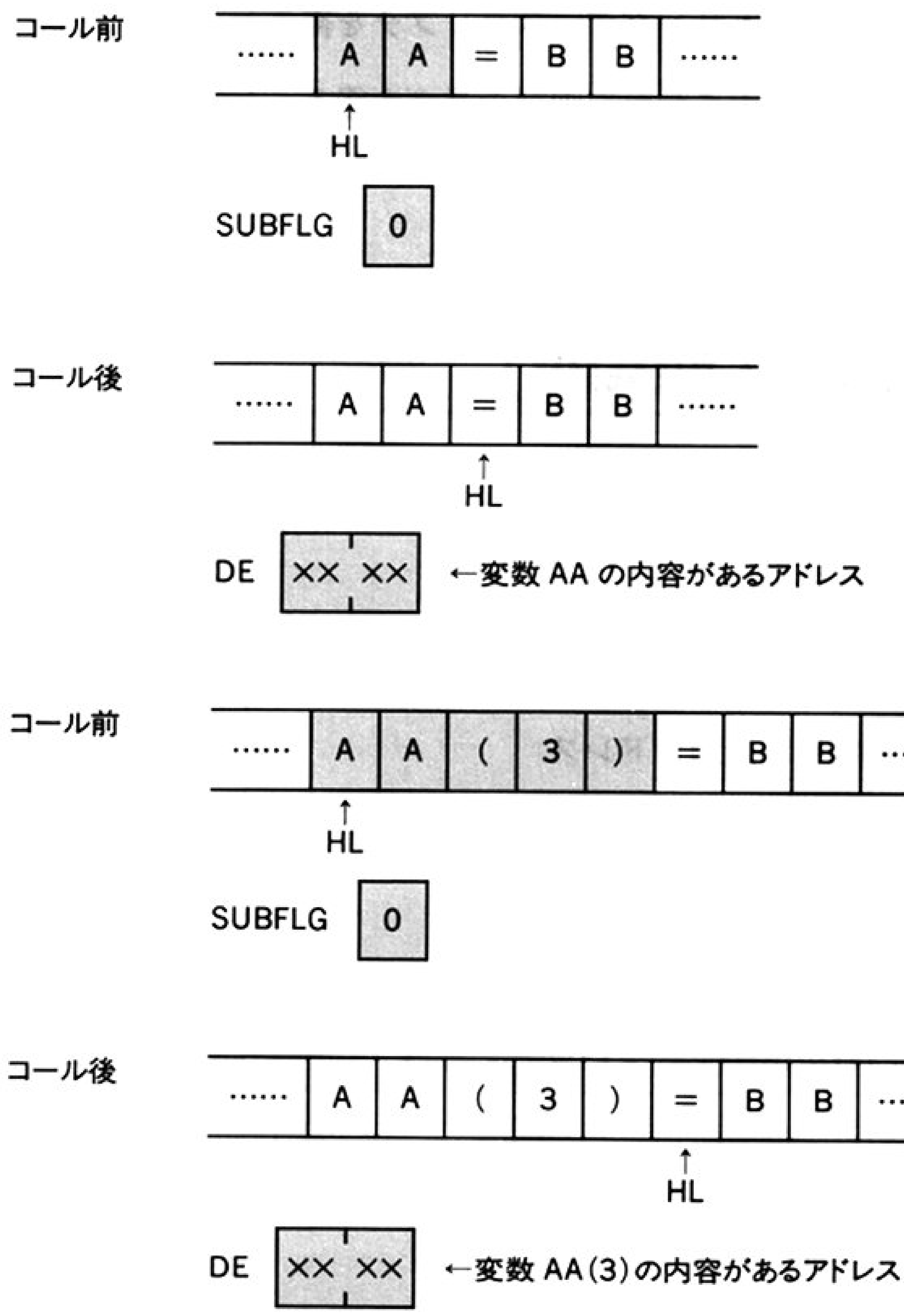
すべて

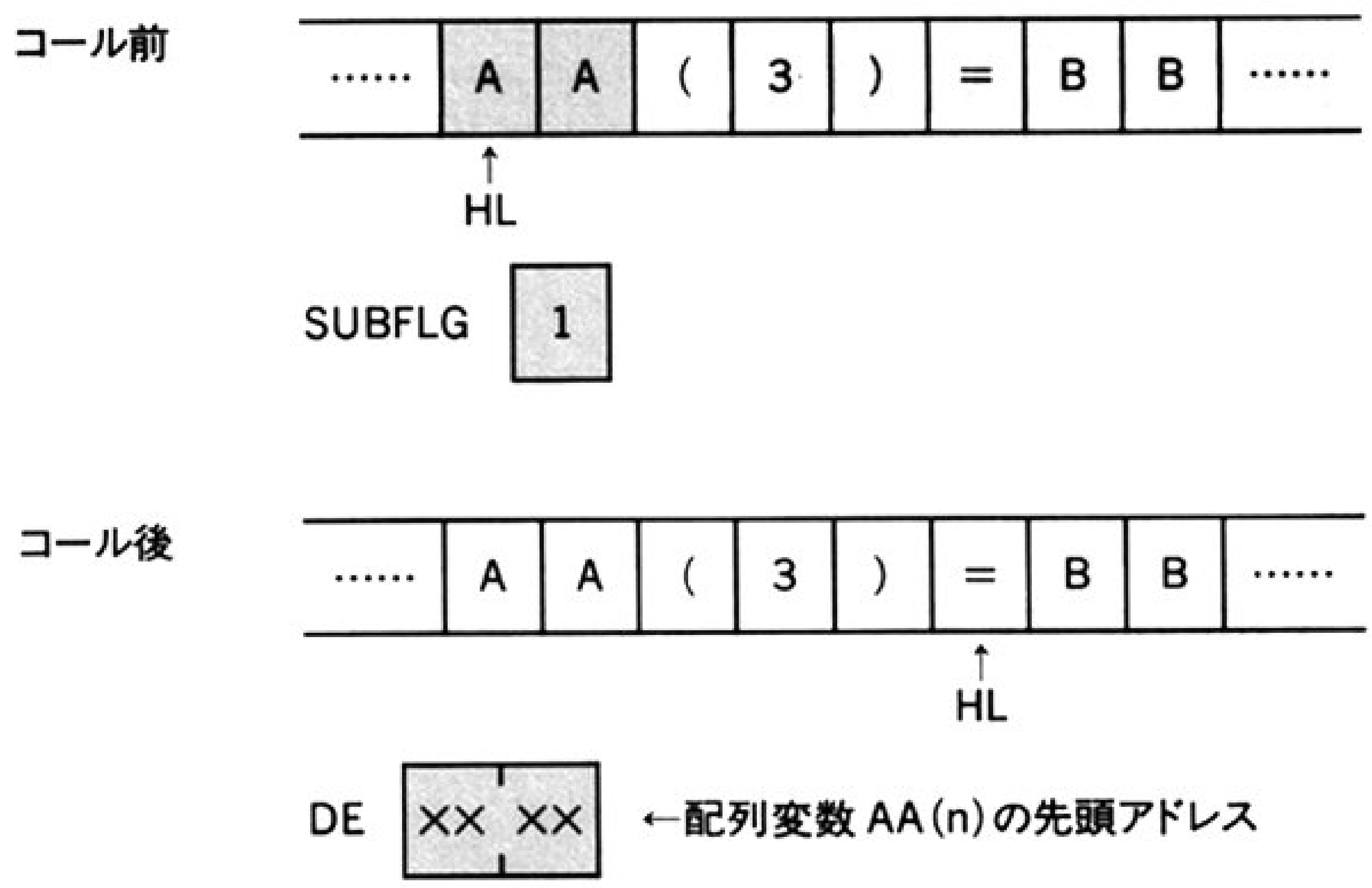
解 説

変数（または配列変数）の格納アドレスを得ます。目的の変数の領域が確保されていなかった場合は、領域の確保も行われます。【SUBFLG(0F6A5H)】の値を 0 以外に設定しておくと配列の個々の要素ではなく、その配列の先頭アドレスが得られます。

BASIC で、USR 関数により実行中の機械語プログラムから、BASIC の変数(ここでは S%）に直接結果を返すには、以下のようにします。

```
vars:
        defd    'S%',0
        ld      hl,vars
        call    ptrget
```





CRUNCH (42B2H / MAIN)

機　能

BASIC のステートメントを中間コードに変換します。

コール手順

HL　　00H で終わる ASCII 形式のテキストの先頭アドレス

戻り値

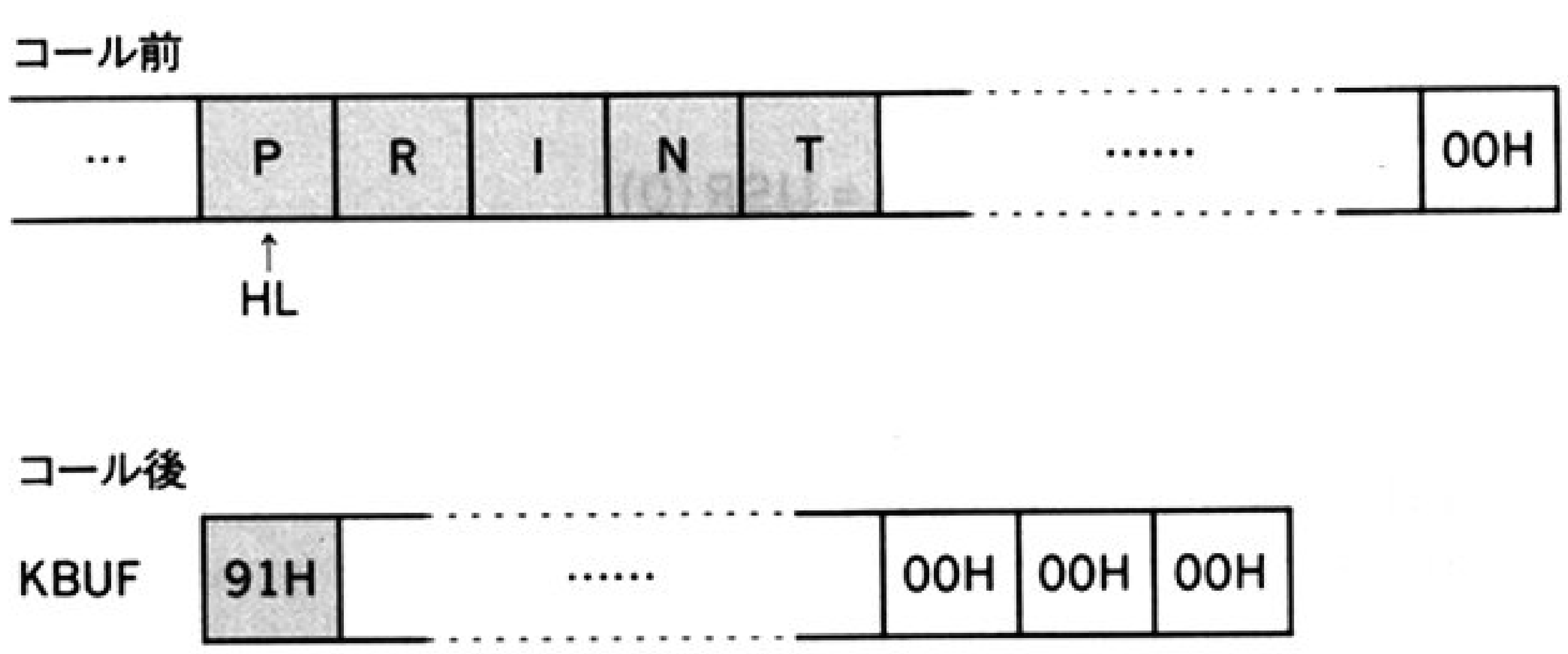
【KBUF(0F41FH)】　　中間コード

変更レジスタ

すべて

解　説

BASIC のステートメントを中間コードに変換します（表 2.17 参照）。



NEWSTT（4601H / MAIN）

機 能

任意のテキストを実行します。

コール手順

HL　　実行するテキストのアドレス

戻り値

なし

変更レジスタ

すべて

解 説

テキストの実行を行います。テキストは下図の状態となっている必要があります。



これらの内部ルーチンは、BASIC テキストを対象とするものであるため、エラーが発生した場合は、BASIC と同様のエラー処理が行われます。その場合、【H.ERRO (0FFB1H)】の書き換えにより、ユーザーがエラー処理を行うこともできます。その場合は、E レジスタにエラー番号が入ります。リスト 2.7 を参照してください。

リスト 2.7　エラー処理ルーチンの変更

```
;
;       エラー処理ルーチンの変更
;       使用法：DEF USER =&Hxxxx：  A = USR(0)
;
herr    equ     0ffb1h          ; error HOOK
synerr  equ     2               ; syntax error code
chput   equ     0a2h            ; character output BIOS
newstt  equ     04601h          ; run
readyr  equ     0409bh

;       エラー処理ルーチンの変更
```

```
start::
        ld      bc,5            ; set new HOOK
        ld      de,herr
        ld      hl,hdat
        ldir
        ret

hdat:
        jp      error
        nop
        nop

 ;      エラー処理ルーチンの実体

error:
        ld      a,e             ; when in error, E holds error code
        cp      synerr          ; syntax error?
        ret     nz              ; no

        ld      hl,data1        ; yes
loop:   ld      a,(hl)          ; put new error message
        cp      '$'
        jr      z,exit
        push    hl
        call    chput
        pop     hl
        inc     hl
        jr      loop
exit:
        jp      readyr

data1:  defm    'マチガッテマス' ; new error start
        defb    07h,07h,07h,'$

        end
```

5.5 拡張ステートメント

4.4 で紹介したのは、「CMD」という未使用の予約語を使って、BASIC の機能を拡張する方法でした。しかし、この方法は予約語を使っている関係で、市販のアプリケーションソフトウェアでは使用できません。ここで紹介する拡張ステートメントは、ROM カートリッジでの使用という条件がつきますが、将来に渡って互換性が保証されている方法ですので、市販のソフトウェアで採用するのに最適です。

拡張ステートメントは CALL 命令で始まり、以下の書式で使用します。

CALL <拡張ステートメント名>[(<引数>[,<引数>...])]

拡張ステートメントは、MSXのカートリッジヘッダーのSTATMENTに拡張ステートメントルーチンのアドレスを書き込むことにより実行できるます。その実現方法などについては、「xxx カートリッジヘッダー」をご参照ください。

添付のフロッピーディスクに、BASICの拡張ステートメントのサンプルプログラムが入っています。ファイル名は「FOO_BAR.MAC」です。このソースプログラムを、

```
M80 = foo_bar
```

でアセンブルして、

```
L80 /P:4000,foo_bar,foo_bar/n/x/e
```

でリンクすると、FOO_BAR.HEXというファイルができます。

このファイルをPROMライターに転送してROMに書き込み、カートリッジにすると「FOO」と「BAR」という拡張ステートメントが使用できるようになります。「FOO」と「BAR」は以下の機能を持っています。

CALL FOO

書　式

CALL FOO(<文字変数>, <文字列>)

機　能

<文字列>を大文字に変換して、<文字変数>に代入します。

文　例

```
CALL FOO(A$,"This is an example")
PRINT A$
```

表示 THIS IS AN EXAMPLE

CALL BAR

書　式

CALL BAR(<数値変数>, <数式1>, <数式2>)

機　能

<数式1>と<数式2>を足し算して、その結果を<数値変数>に代入します。

文　例

```
CALL BAR(A,100,23)
PRINT A
```

表示 123

6章 内部ルーチン

6.1 PLAY 文 BIOS

6.1.1 概要

MSX は PSG の音楽を割り込みルーチン内で演奏しています。BASIC では PLAY 文が MML (Music Macro Language) を割り込みルーチンが解釈できる形に変換してキューに書き込み、あるフラグをセットします。割り込みルーチンはそのフラグを見て、キューにデータが書かれたことを知り、キューからデータを取り出して演奏します。

演奏はキューから取り出したデータが END MARK のときに終了します。このキューは各チャンネルごとに 1 本ずつで、合計 3 本あります。

6.1.2 キューの構造

まず、【QUEUES(0F3F3H,2)】というワークエリアがあり、これが QCB(Queue Control Block) という構造体の配列の先頭を指しています。QCB 構造体の配列は、通常【QUETAB(0F959H,24)】にあり、要素数は 4 つです。しかし、最後の 1 つは PLAY 文では使用しません。1 つの QCB は次のような構造をしています。

表 2.19 QCB の構造

開始番地からのオフセット	用途
0	PUT オフセット　キューにデータを書き込むときのオフセット
1	GET オフセット　キューからデータを読み出すときのオフセット
2	バックアップキャラクタ キューにデータを書き込むときのに一時的に使われる
3	キューの長さ　実際のキューの長さ－1。取り得る値は2のn乗－1(nは2から8)。単位はバイト。通常は127
4、5	キューの開始番地 実際のキューの開始番地。通常、以下のキューを指している チャンネル1　【VOICAQ(0F975H,128)】 チャンネル2　【VOICBQ(0F9F5H,128)】 チャンネル3　【VOICCQ(0FA75H,128)】

6.1.3 キューデータの解説

キューには1回の割り込み処理中に設定されるデータが可変長で入っています。最初の2バイトの上位3ビットはその割り込み処理中に処理されるデータが何バイトであるかを指定します。指定できる範囲は1から5です。残りの13ビットは次のデータの処理をするまでの時間をタイマー割り込みの回数を指定します。この2バイトのデータは上位バイトが先になります。終了はこの2バイトの代わりに、1バイトのFFHが入ります。残りのデータは次の4種類のうちのどれかが入っているかで異なります。

1. 音程を設定するデータ（MMLのAからG、Nに相当する）
2. 音程を設定するデータ（Vに相当する）
3. エンベロープの形状を設定するデータ（Sに相当する）
4. エンベロープの周期を設定するデータ（Mに相当する）

各々の詳細な形式を以下で説明します。PSGに設定するデータの詳細は、「第xx部 x章 PSG」を参照してください。

1. 音程を設定するデータ

音程を設定するデータは2バイトで構成されます。上位2ビットは音程を設定するデータであ

ることを示すために、「00」とします。残りの14ビットでPSGに設定する音色をそのまま指定しますが、PSGでは下位12ビットのみ有効です。上位バイトが先になります。

2. 音量を設定するデータ

音量を設定するデータは1バイトで構成されます。上位4ビットは音量を設定するデータであることを示すために、「1000」とします。残りの4ビットでPSGに設定する音量をそのまま指定します。

3. エンベロープの形状を設定するデータ

エンベロープの形状を設定するデータは1バイトで構成されます。上位4ビットはエンベロープの形状を設定するデータであることを示すために、「1001」とします。残りの4ビットでPSGに設定するエンベロープの形状をそのまま指定します。

4. エンベロープの周期を設定するデータ

エンベロープの周期を設定するデータは3バイトで構成されます。最初の1バイトはエンベロープの周期を設定するデータであることを示すために「40H」とします。残りの2バイトでPSGに設定するエンベロープの周期をそのまま指定します。この2バイトは上位バイトが先になります。

5. エンベロープの形状と周期を同時に設定するデータ

このデータは3バイトで構成されます。最初の1バイトのうち、上位4ビットは「1101」とします。下位4ビットはPSGに設定するエンベロープの形状をそのまま指定します。残りの2バイトでPSGに設定するエンベロープの周期をそのまま指定します。この2バイトは上位バイトが先になります。

6.1.4 データの具体例

MMLで"T120L4V8O4CDE"とした場合（以下16進数）

60,1E	3バイトのデータがあり、0.5秒間続ける
88	音量は8
01,AC	音程は01ACH
60,1E	3バイトのデータがあり、0.5秒間続ける
88	音量は8

01,7D　音程は 017DH

60,1E　3 バイトのデータがあり、0.5 秒間続ける

88　音量は 8

01,53　音程は 0153H

FF　終了

6.1.5 BIOS

BIOS は STRTMS(0099H / MAIN) という名前のルーチンがあります。これを呼ぶことにより、演奏を始めることができます。以下のサンプルプログラムを参照して下さい。

リスト 2.8　STRTMS の使用例

```
        strtms  equ     0099h
        musicf  equ     0fb3fh      ; 音楽演奏用の割り込みフラグ
        plycnt  equ     0fb40h      ; キューされている PLAY 文の数
        quetab  equ     0f959h      ; キューテーブル
        qcblen  equ     6           ; QCB の長さ
        cseg
music:
        ld      a,(musicf)          ; 演奏中?
        or      a
        jr      nz,music            ; 演奏中なら、終わるまで待つ
        di
        ld      de,quetab
        ld      hl,queini
        ld      bc,qcblen * 3
        ldir                        ; QCB の設定
        ld      a,1
        ld      (plycnt),a
        call    strtms              ; 演奏開始
        ei
        ret
queini:
        db      0ffh
        db      0
        db      0
        db      0ffh
        dw      queue1
        db      0ffh
        db      0
        db      0
        db      0ffh
        dw      queue2
        db      0ffh
        db      0
        db      0
        db      0ffh
```

```
        dw      queue3
;
; "T120L4V8O4CDE"
;
queue1:
        db      0                       ; a dummy
        db      60h,1eh,88h,01h,0ach    ; T120L4V8C
        db      60h,1eh,88h,01h,7dh     ; T120L4V8D
        db      60h,1eh,88h,01h,53h     ; T120L4V8E
        db      0ffh                    ; end mark
queue2:
        db      0
        db      0ffh
queue3:
        db      0
        db      0ffh

        end
```

6.2 ビットブロックトランスファ

ビットブロックトランスファとは、RAM、VRAM、ディスク間でデータを転送する SUB ROM 内ルーチンです。本来、MSX2 で拡張された BASIC の COPY 命令用のルーチンですが、完全なサブルーチンの形式になっているので、BIOS のようにユーザープログラムから利用できます。

なお、ビットブロックトランスファルーチンが使用可能な画像データは、BASIC の COPY 命令で作成されたデータです。BSAVE "ファイル名",&Hxxxx,&Hxxxx,S で作成されたデータは使用できません。

X 方向のサイズ	Y 方向のサイズ	VRAM データ

図 2.28　COPY 命令で作成した画像ファイルのヘッダ情報

例

COPY (0,0)-(10,20) TO "TEST.PIC"

で作成した TEST.PIC は、0BH、00H、15H、00H、 .... となる。

6.2.1 VRAM間の転送

BLTVV(0191H / SUB)

機　能

VRAM 領域内で転送を行います。

コール手順

HL　　0F562H

さらに以下のパラメータを設定する。

SX(0F562H,2)	；転送元基準点　X座標
SY(0F564H,2)	；転送元基準点　Y座標
DX(0F566H,2)	；転送先基準点　X座標
DY(0F568H,2)	；転送先基準点　Y座標
NX(0F56AH,2)	；X方向転送ドット数
NY(0F56CH,2)	；Y方向転送ドット数
CDUMMY(0F56EH,1)	；ダミー（セットする必要はない）
ARG(0F56FH,1)	；方向と拡張RAMの選択（VDPのR#45と同じ）
LOGOP(0F570H,1)	；ロジカルオペレーションコード

戻り値

CYフラグをリセット

変更レジスタ

すべて

例

```
; COPY (0,0)-(255,99) TO (0,100)

bltvv   equ     0191h
extrom  equ     015fh
        ld      hl,0
        ld      (sx),hl         ; 転送元基準点は (0,0)
        ld      (sy),hl
        ld      (dx),hl         ; 転送先基準点は (0,100)
        ld      l,100
        ld      (dy),hl
        ld      hl,255-0+1      ; Y方向転送ドット数の設定
```

```
ld      (nx),hl
ld      hl,99−0+1       ; X 方向転送ドット数の設定
ld      (ny),hl
xor     a               ; 方向の設定
ld      (arg),a
ld      (l_op),a        ; operation is 'replace'
                        ;
                        ;
ld      hl,0f562h
ld      ix,bltvv
call    extrom
        :
        :
```

6.2.2 RAMとVRAM間のデータ転送

以下のルーチンを使う場合、グラフィックのデータ領域として、スクリーンモードによって次のような大きさの領域を確保しておく必要があります。

表 2.20 スクリーンモード別確保領域の算出法

スクリーンモード	領域の大きさ
6	X 方向のドット数×Y 方向のドット数÷4+4
5、7	X 方向のドット数×Y 方向のドット数÷2+4
8	X 方向のドット数×Y 方向のドット数+4

注 意　割り算の端数は切り上げて下さい。

なお、ディスクやRAMでは配列データと同様にサイズを示すデータが付加されます。このデータは先頭から2バイトがX方向のドット数、次の2バイトがY方向のドット数を示しています。

BLTVM(0195H / SUB)

機 能

配列をVRAMに転送します。

コール手順

HL　　0F562H

さらに以下のパラメータを設定する。

DPTR(0F562H,2)	; メモリの転送元アドレス
DUMMY(0F564H,2)	; ダミー（セットする必要はない。）
DX(0F566H,2)	; 転送先基準点　X 座標
DY(0F568H,2)	; 転送先基準点　Y 座標
NX(0F56AH,2)	; X 方向転送ドット数（セットする必要はない）
NY(0F56CH,2)	; Y 方向転送ドット数（セットする必要はない）
CDUMMY(0F56EH,1)	; ダミー（セットする必要はない）
ARG(0F56FH,1)	; 方向と拡張 RAM の選択（VDP の R#45 と同じ）
LOGOP(0F570H,1)	; ロジカルオペレーションコード

戻り値

転送するデータの個数がおかしければ CY フラグをセット

変更レジスタ

すべて

例

```
; COPY <B000H> TO (5,6)

bltvm   equ     0195h
extrom  equ     015fh
        ld      hl,0b000h
        ld      (dptr),hl       ; データは B000H にある
        ld      hl,5
        ld      (dx),hl         ; 転送先基準点は (5,6)
        inc     hl
        ld      (dy),hl
        xor     a               ; 方向の設定
        ld      (arg),a
        ld      (l_op),a        ; operation is 'replace'
                                ;
        ld      hl,0f562h
        ld      ix,bltvm
        call    extrom
                :
```

BLTMV(0199H / SUB)

機 能

VRAMから配列に転送します。

コール手順

HL　　0F562H

さらに以下のパラメータを設定する

SX(0F562H,2)	；	転送元基準点　X座標
SY(0F564H,2)	；	転送元基準点　Y座標
DPTR(0F566H,2)	；	メモリの転送元アドレス
DUMMY(0F568H,2)	；	ダミー（セットする必要はない）
NX(0F56AH,2)	；	X方向転送ドット数
NY(0F56CH,2)	；	Y方向転送ドット数
CDUMMY(0F56EH,1)	；	ダミー（セットする必要はない）
ARG(0F56FH,1)	；	方向と拡張RAMの選択（VDPのR#45と同じ）

戻り値

CYフラグをリセット

変更レジスタ

すべて

例

```
; COPY (10,20)-(50,25) TO <B000H>

bltmv    equ    0199h
extrom   equ    015fh
         ld     hl,0b000h
         ld     (dptr),hl        ; データはB000Hにある
         ld     hl,10
         ld     (sx),hl          ; 転送基準点は(10,20)
         ld     hl,20
         ld     (sy),hl
         ld     hl,50-10+1       ; X方向転送ドット数の設定
         ld     (nx),hl
         ld     hl,25-20+1       ; Y方向転送ドット数の設定
         ld     (ny),hl
         xor    a                ; 方向の設定
         ld     (arg),a
                                 ;
         ld     hl,0f562h
         ld     ix,bltmv
```

```
                call    extrom
                          :
```

6.2.3 ディスクとRAM、VRAM間の転送

ディスクを使う場合、まずファイル名を設定する必要があります(ファイル名はBASICと同様に指定)。例を次に示します。

```
                :
        ld      hl,fname        ; ファイル名へのポインタの獲得
        ld      (fnptr),hl      ; パラメータエリアに設定
                :
fname : db 22h,'b:test.pic', 22h, 0     ; ''TEST.PIC'', end mark
```

ここで、何かエラーが起きれば、BASICインタープリタのエラーハンドラーへジャンプします。そのため、自分のプログラム内でエラーをハンドリングしたい場合やMSX-DOS、ROMカートリッジでこのルーチンを呼ぶ場合、フックをセットする必要があります。このフックは【H.ERRO(0FFB1H)】です。

BLTVD(019DH / SUB)

機 能

ディスクからVRAMへ転送します。

コール手順

HL　　0F562H

さらに以下のパラメータを設定する

FNPTR(0F562H,2)	; ファイル名のあるアドレス
DUMMY(0F564H,2)	; ダミー(セットする必要はない)
DX(0F566H,2)	; 転送先基準点　X座標
DY(0F568H,2)	; 転送先基準点　Y座標
NX(0F56AH,2)	; X方向転送ドット数(セットする必要はない)
NY(0F56CH,2)	; Y方向転送ドット数(セットする必要はない)

CDUMMY(0F56EH,1)　　；ダミー（セットする必要はない）
ARG(0F56FH,1)　　；方向と拡張RAMの選択（VDPのR#45と同じ）

戻り値

CYフラグをリセット

変更レジスタ

すべて

例

```
; COPY "TEST.PIC" TO (3,2)

bltvd     equ     019dh
extrom    equ     015fh
          ld      hl,filename
          ld      (fnptr),hl       ; ファイル名へのポインタの設定
          ld      hl,3
          ld      (dx),hl          ; 転送先基準点は(3,2)
          dec     hl
          ld      (dy),hl
          xor     a                ; 方向の設定
          ld      (arg),a
          ld      (l_op),a         ; operation is 'replace'
                                   :
          ld      hl,0f562h
          ld      ix,bltvd
          call    extrom
                  :
filename: db    22h,'test.pic',22h,0
                  :
```

BLTDV(01A1H / SUB)

機　能

VRAMからディスクへ転送します。

コール手順

HL　　0F562H
さらに以下のパラメータを設定する

FNPTR(0F562H,2)　　；ファイル名のあるアドレス

DUMMY(0F564H,2)　　；ダミー（セットする必要はない)
DX(0F566H,2)　　；転送先基準点　X 座標
DY(0F568H,2)　　；転送先基準点　Y 座標
NX(0F56AH,2)　　；X 方向転送ドット数
NY(0F56CH,2)　　；Y 方向転送ドット数
CDUMMY(0F56EH,1)　　；ダミー（セットする必要はない)
ARG(0F56FH,1)　　；方向と拡張 RAM の選択（VDP の R#45 と同じ)

戻り値

CY フラグをリセット

変更レジスタ

すべて

例

```
; COPY (10,20)-(50,25) TO "A:TEST.PIC"

bltdv       equ     01a1h
extrom      equu    015fh
            ld      hl,filename
            ld      (fnptr),hl      ; ファイル名へのポインタの設定
            ld      hl,10
            ld      (sx),hl         ; 転送先基準点は(10,20)
            ld      hl,20
            ld      (sy),hl
            ld      hl,50-10+1      ; X 方向転送ドット数の設定
            ld      (nx),hl
            ld      hl,25-20+1      ; Y 方向転送ドット数の設定
            ld      (ny),hl
            xor     a               ; 方向の設定
            ld      (arg),a
                                    ;
            ld      hl,0f562h
            ld      ix,bltdv
            call    extrom
                    :
filename:db     22h,'a:test.pic',22h,0
                    :
```

BLTMD(01A5H / SUB)

機　能

ディスクから配列データをロードします。

コール手順

HL　　0F562H

さらに以下のパラメータを設定する

FNPTR(0F562H,2)	；	ファイル名のあるアドレス
SY(0F564H,2)	；	ダミー（セットする必要はない）
SPTR(0F566H,2)	；	ロードする先頭アドレス
EPTR(0F568H,2)	；	ロードする最終アドレス

戻り値

CY フラグをリセット

変更レジスタ

すべて

例

```
bltmd    equ   01a5h
extrom   equ   015fh
         ld    hl,filename      ; ファイル名へのポインタの設定
         ld    (fnptr),hl
         ld    hl,8000h         ; 先頭アドレスの設定
         ld    (sptr),hl
         ld    hl,0afffh        ; 最終アドレスの設定
         ld    (eptr),hl
                                ;
         ld    hl,0f562h
         ld    ix,bltmd
         call  extrom
               :
filename:db  22h,'test.bin',22h,0
               :
```

BLTDM(01A9H/SUB)

機　能

ディスクへ配列データをセーブします。

コール手順

HL　　0F562H

さらに以下のパラメータを設定する

SPTR(0F562H,2)　　；セーブする先頭アドレス
EPTR(0F564H,2)　　；セーブする最終アドレス
FNPTR(0F566H,2)　　；ファイル名のあるアドレス

戻り値

CY フラグをリセット

変更レジスタ

すべて

例

```
bltdm      equ     01a9h
extrom     equ     015fh
           ld      hl,filename       ; ファイル名へのポインタの設定
           ld      (fnptr),hl
           ld      hl,8000h          ; 先頭アドレスの設定
           ld      (sptr),hl
           ld      hl,0afffh         ; 最終アドレスの設定
           ld      (eptr),hl
                                     ;
           ld      hl,0f562h
           ld      ix,bltdm
           call    extrom
                   :
filename:db        22h,'a:test.bin',22h,0
                   :
```

6.2.4 ビットブロックトランスファの利用法

1. BASICの環境から利用する場合

定められたワークエリアに値を入れて、HLレジスタが0F562Hを指すようにセットし、SUB ROM内のアドレスをコールします。

以下のサンプルプログラムは、「MSX-DOS TOOLS」を使用し、BASICの環境からマシン語のプログラムをメモリ上にロードし、実行するものです。

リスト2.9 BASIC環境でのビットブロックトランスファ使用例

```
;
;           BLTVV.MAC   sample progrem
;
;
bltvv       equu    0191h
sx          equ     0f562h
sy          equ     0f564h
dx          equ     0f566h
dy          equ     0f568h
nx          equ     0f56ah
ny          equ     0f56ch
cdummy      equ     0f56eh
arg         equ     0f56fh
logop       equ     0f570h
extrom      equ     015fh
start::
            ld      de,0
            ld      (sx),de         ; 転送元基準点は (0,0)
            ld      (sy),de
            ld      (dx),de         ; 転送先基準点は (0,100)
            ld      de,100
            ld      (dy),de
            ld      (nx),de         ; X方向転送ドット数を100に設定
            ld      (ny),de         ; Y方向転送ドット数を100に設定
            ld      a,00000000b     ; VRAMに対してコピーする
            ld      (arg),a         ; 方向は、X方向が右、Y方向が下
            ld      (logop),a       ; ロジカルオペレーションはIMP
            ld      hl,sx           ; hlレジスタにF562Hをセット
            ld      ix,bltvv
            call    extrom          ; SUB ROMコール
            ret

            end
```

以下のバッチファイルは、上記のアセンブラプログラムをBASICからBLOADするバイナリプログラムにするものです。

```
m80 = bltvv
l80 /p:c000,bltvv,bltvv/n/x/e:start
bsave bltvv.hex > bltvv.bin
```

こうして作成したバイナリファイルを実行するには、以下のBASICプログラムを使用します。

```
10 CLEAR 300,&HBFFF
20 BLOAD"bltvv.bin"
30 SCREEN 8:COPY "sample.pic" TO (0,0)
40 DEFUSR0 =&HC000
50 A = USR(0)
60 A$= INKEY$:IF A$="" THEN GOTO 60
```

2. MSX-DOSの環境から利用する場合

ビットブロックトランスファルーチンは、本来BASICのCOPY命令で使用するものですから、BASIC環境での動作を前提に作られています。したがって、MSX-DOSのプログラムからビットブロックトランスファを利用する場合には、BASIC環境からアクセスする手順に加えて、以下の手続きが必要になります。

1) ビットブロックトランスファルーチンは、SPと【STREND(0F6C6H)】が指す番地までをディスクアクセス時のバッファとして使用するので、STRENDに値を設定しなければならない。

2) ディスクをアクセスするルーチン使用時に設定するFNAMEの内容は、ページ2かページ3になければならない。

リスト2.10 DOS環境でのビットブロックトランスファ使用例

```
;
;  BLTVD sample progrem
;
strend  equ     0f6c6h
bltvd   equ     019dh
calslt  equ     001ch
exptbl  equ     0fcc1h
exbrsa  equ     0faf8h
fnptr   equ     0f562h
dx      equ     0f566h
dy      equ     0f568h
arg     equ     0f56fh
l_op    equ     0f570h
chgmod  equ     005fh
        extrn   _calsub
;
start:
        ld      sp,(6)          ; スタックを設定
```

```
        ld      hl,0c000h       ; STRENDをC000Hに設定
        ld      (strend),hl
        ld      bc,11           ; FNAMEの内容をB000Hからにブロック転送
        ld      de,0b000h
        ld      hl,fname
        ldir
        ld      a,8             ; スクリーン8
        ld      iy,(exptbl-1)
        ld      ix,chgmod
        call    calslt
        ld      hl,0b000h       ; ブロック転送したアドレスを設定
        ld      (fnptr),hl
        ld      de,0
        ld      (dx),de
        ld      (dy),de
        xor     a

        ld      (arg),a
        ld      (l_op),a
        ld      hl,0f562h
        ld      ix,bltvd
        call    _calsub

        ld      a,0             ; スクリーンをテキストモードに戻す
        ld      iy,(exptbl-1)
        ld      ix,chgmod
        call    calslt
        jp      0
;
fname:  db      22h,'test.pic',22h,0
;
        end
```

6.3 Math-Pack

Math-Pack とは、MSX BASIC の数値演算ルーチンの中核をなすもので、これらをマシン語から呼び出すことにより、簡単に浮動小数点演算や三角関数ルーチンを利用することができます。

Math-Pack での実数演算はすべて BCD (Binary Coded Decimal = 2 進化 10 進法) 表現によって行われます。また、実数表現には、「単精度」と「倍精度」の 2 種類があり、単精度（6 桁）は 4 バイト、倍精度（14 桁）は 8 バイトで表現されます。(図 2.29、図 2.30)

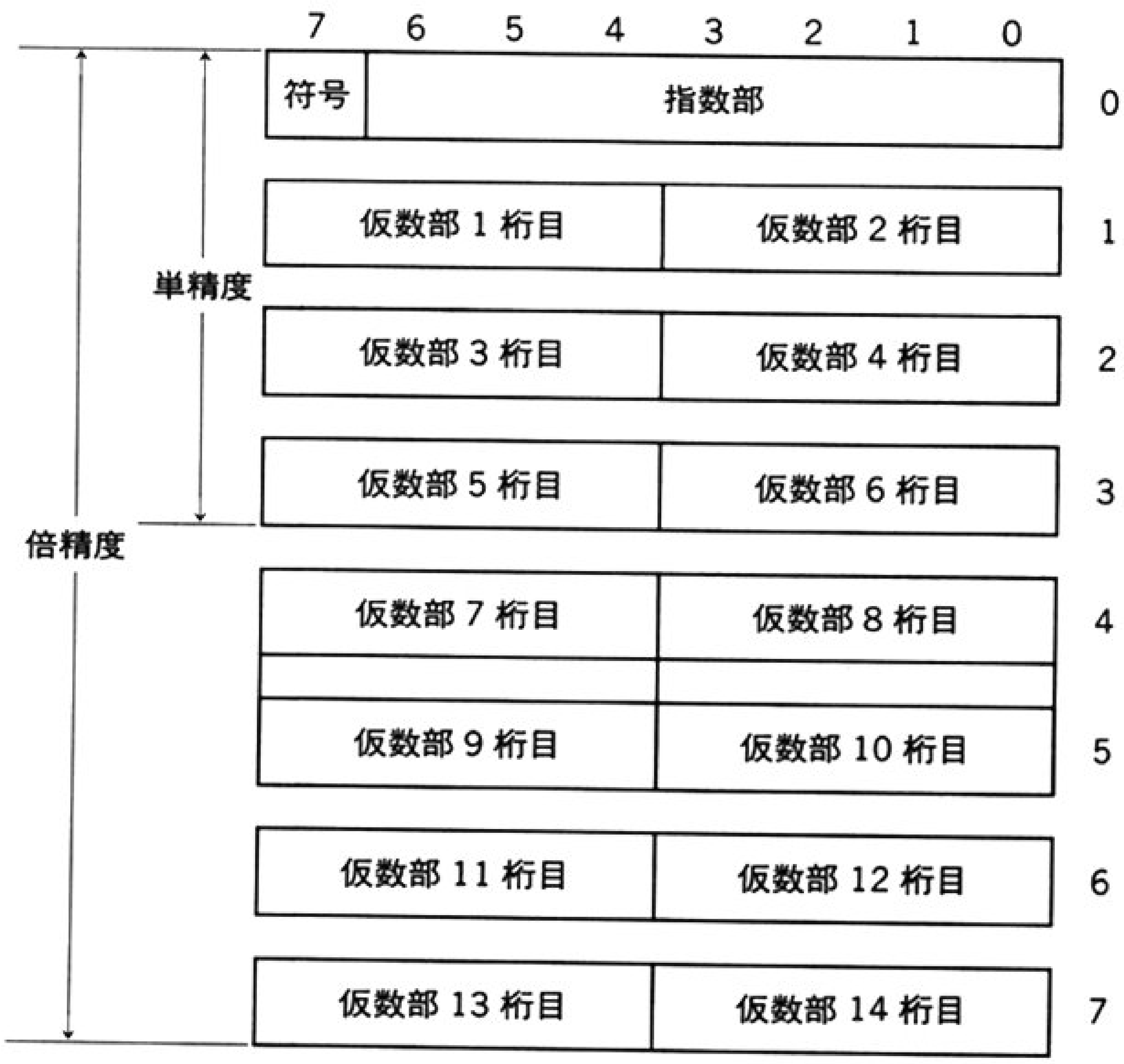


図 2.29 BCD による実数表現のフォーマット

単精度表現の例

123456 → 0.123456 E+6

	0	1	2	3
DAC	46	12	34	56

倍精度表現の例

123456.78901234 → 0.12345678901234 E+6

	0	1	2	3	4	5	6	7
DAC	46	12	34	56	78	90	12	34

図 2.30 実数の表現例

1つの実数は、符号部、指数部、仮数部から成り立っています。符号部とはこの仮数部の符号を表し、0ならば正、1ならば負となります。指数部はバイナリ表現であり、+63乗〜−63乗まで表すことができます（図2.31）。倍精度実数の有効範囲を図2.32に示します。

符号	指数部							意味
0	0	0	0	0	0	0	0	………0
1	0	0	0	0	0	0	0	………未定義(−0?)
×	0	0	0	0	0	0	1	………10の−63乗
×	1	0	0	0	0	0	0	………10の0乗
×	1	1	1	1	1	1	1	………10の+63乗

注 意　「×」はその値が1または0のどちらであっても関係がないことを表します。

図 2.31 指数部フォーマット

	0	1	2	3	4	5	6	7	(バイト)
DAC	FF	99	99	99	99	99	99	99	−0.99999999999999 E+63
					⋮				⋮
	81	10	00	00	00	00	00	00	−0.10000000000000 E−63
	00	×	×	×	×	×	×	×	0
	01	10	00	00	00	00	00	00	+0.10000000000000 E−63
					⋮				⋮
DAC	7F	99	99	99	99	99	99	99	+0.99999999999999 E+63

図 2.32 倍精度実数表現の有効範囲

Math-Pack では、演算の対象となるメモリがあらかじめ決められています。このメモリエリアを【DAC(Decimal ACcumlator - 0F7F6H)】と呼び、DAC との演算対象となる数値を格納するエリアを【ARG(0F847H)】といいます。例えば、乗算ルーチンの場合、DAC に入っている数と ARG に入っている数の積が計算され、その結果が DAC に入ることになります。

DAC には、単精度実数、倍精度実数、2バイト整数を格納することができますが、これらのうちのどれが入っているのかを示すために【VALTYP(0F663H)】が使われ、それぞれの場合について 4、8、2 の値が入ります。

単精度実数と倍精度実数は、いずれも DAC の先頭から数値を格納しなければなりません。ただし、2バイト整数の場合だけは、下位、上位を DAC+2、+3 に格納してください。

演算の対象が明記されていない演算は【VALTYP (0F663H)】によって値の型を参照するので、呼び出す前に正しい型を設定して下さい。

Math-Pack は BASIC の内部ルーチンであるため、エラーが起きた場合 (0除算やオーバーフローなど) はそれに応じたエラールーチンへ自動的に分岐し、その後 BASIC のコマンドレベルに戻ります。それを避けたいときは、【H.ERRO(0FFB1H)】を書き換えて、アプリケーションプログラム側のエラー処理ルーチンへジャンプするようにして下さい。

6.3.1 Math-Pack ワークエリア

ラベル	アドレス	サイズ	意　味
VALTYP	F663H	1	DACに入っている数の形式を表す
DAC	F7F6H	16	BCD形式の浮動小数点アキュムレータ
ARG	F847H	16	DACの演算対象

6.3.2 Math-Pack エントリ

1. 基本演算

ラベル	アドレス	意　味	変化するレジスタ
DECSUB	268CH	DAC ← DAC-ARG	すべて
DECADD	269AH	DAC ← DAC+ARG	すべて
DECNRM	26FAH	DACを正規化する*1	すべて
DECROU*2	273CH	DACを四捨五入する	すべて
DECMUL	27E6H	DAC ← DAC×ARG	すべて
DECDIV	289FH	DAC ← DAC÷ARG	すべて

注　意　これらは、いずれもDAC、ARGにある数を倍精度実数として扱う。

*1　仮数部の不要な0を取り去ること（0.00123 → 0.123E-2）

*2　DECROUは倍精度の値の最終桁+1（15桁目）に対して演算を行う。

<演算の例>

1.23456789012345 → DECROU → 1.2345678901235
1.11111111111114 → DECROU → 1.1111111111111
1.11111111111115 → DECROU → 1.1111111111112
1.99999999999999 → DECROU → 2.0

2. 関数1

ラベル	アドレス	意　味	変化するレジスタ
COS	2993H	DAC←COS(DAC)	すべて
SIN	29ACH	DAC←SIN(DAC)	すべて
TAN	29FBH	DAC←TAN(DAC)	すべて
ATN	2A14H	DAC←ATN(DAC)	すべて
LOG	2A72H	DAC←LOG(DAC)	すべて
SQR	2AFFH	DAC←SQR(DAC)	すべて
EXP	2B4AH	DAC←EXP(DAC)	すべて
RND	2BDFH	DAC←RND(DAC)	すべて
INT	30CFH	DAC←INT(DAC)	すべて

注　意　すべて、BASICにおける同名の関数の処理ルーチン。
変化するレジスタすべてとは、A、B、C、D、E、H、Lの各レジスタを指す。

3. 関数2

ラベル	アドレス	意　味	変化するレジスタ
SIGN	2E71H	A←DACの符号	A
ABSFN	2E82H	DAC←ABS(DAC)	すべて
NEG	2E8DH	DAC←NEG(DAC)	A、H、L
SGN	2E97H	DAC←SGN(DAC)	A、H、L

4. 移動

ラベル	アドレス	意　味	対　象	変化するレジスタ
MAF	2C4DH	ARG←DAC	倍精度	A、B、D、E、H、L
MAM	2C50H	ARG←[HL]	倍精度	A、B、D、E、H、L
MOV8DH	2C53H	[DE]←[HL]	倍精度	A、B、D、E、H、L
MFA	2C59H	DAC←ARG	倍精度	A、B、D、E、H、L
MFM	2C5CH	DAC←[HL]	倍精度	A、B、D、E、H、L
MMF	2C67H	[HL]←DAC	倍精度	A、B、D、E、H、L
MOV8HD	2C6AH	[HL]←[DE]	倍精度	A、B、D、E、H、L
XTF	2C6FH	[SP]←→DAC	倍精度	A、B、D、E、H、L
PHA	2CC7H	ARG→[SP]	倍精度	A、B、D、E、H、L

ラベル	アドレス	意　味	対　象	変化するレジスタ
PHF	2CCCH	DAC→[SP]	倍精度	A、B、D、E、H、L
PPA	2CDCH	[SP]→ARG	倍精度	A、B、D、E、H、L
PPF	2CE1H	[SP]→DAC	倍精度	A、B、D、E、H、L
PUSHF	2EB1H	DAC→[SP]	単精度	D、E
MOVFM	2EBEH	DAC←[HL]	単精度	B、C、D、E、H、L
MOVFR	2EC1H	DAC←(CBED)	単精度	D、E、
MOVRF	2ECCH	(CBED)←DAC	単精度	B、C、D、E、H、L
MOVRMI	2ED6H	(CBED)←[HL]	単精度	B、C、D、E、H、L
MOVRM	2EDFH	(BCDE)←[HL]	単精度	B、C、D、E、H、L
MOVMF	2EE8H	[HL]←DAC	単精度	A、B、D、E、H、L
MOVE	2EEBH	[HL]←[DE]	単精度	B、C、D、E、H、L
VMOVAM	2EEFH	ARG←[HL]	VALTYP	B、C、D、E、H、L
MOVVFM	2EF2H	[DE]←[HL]	VALTYP	B、C、D、E、H、L
VMOVE	2EF3H	[HL]←[DE]	VALTYP	B、C、D、E、H、L
VMOVFA	2F05H	DAC←ARG	VALTYP	B、C、D、E、H、L
VMOVFM	2F08H	DAC←[HL]	VALTYP	B、C、D、E、H、L
VMOVAF	2F0DH	ARG←DAC	VALTYP	B、C、D、E、H、L
VMOVMF	2F10H	[HL]←DAC	VALTYP	B、C、D、E、H、L

注　意　[HL]、[DE]は、それぞれHL、DEレジスタが指したメモリ上の数値を示す。
()内の4レジスタ名は、左から（符号＋指数部）、(仮数部1、2桁)、(仮数部3、4桁)、(仮数部5、6桁）を示す単精度実数である。
移動の対象がVALTYPとなっているものは、【VALTYP(F663H)】に示されている型に応じた移動（2、4、8バイト）を行う。

5. 比較

ラベル	アドレス	対　象	左辺	右辺	変化するレジスタ
FCOMP	2F21H	単精度実数	CBED	DAC	HL
ICOMP	2F4DH	2バイト整数	DE	HL	HL
XDCOMP	2F5CH	倍精度実数	ARG	DAC	すべて

注　意　結果はすべてAレジスタにはいる。レジスタの意味は、
A＝1→左辺＜右辺
A＝0→左辺＝右辺
A＝-1→左辺＞右辺

単精度の比較で、左辺のCBEDとは各レジスタに、それぞれ単精度の（符号＋指数部）、（仮数部1、2桁）、（仮数部3、4桁）、（仮数部5、6桁）が入ることを意味する。

6. 浮動小数点入出力

ラベル	アドレス	機　能
FIN	3299H	浮動小数点を表す文字列を実数表現に変換して、DACに格納する。

コール手順

HL　文字列の先頭アドレス

戻り値

DAC　実数

C　FFH＝小数点なし、0＝小数点あり

B　小数点以降の桁数

D　総桁数

ラベル	アドレス	機　能
FOUT	3225H	DACに入っている実数表現の数を文字列に変換する（フォーマット指定なし）
PUFOUT	3426H	DACに入っている実数表現の数を文字列に変換する（フォーマット指定あり）

コール手順

A　変換のフォーマット

bit7　0＝フォーマット指定なし、1＝フォーマット指定あり

bit6　0＝カンマ（,）による区切りなし、1＝3ごとにカンマ（,）で区切る

bit5　0＝意味なし、1＝先頭のスペースを「*」で埋める

bit4　0＝意味なし、1＝数値の前に「$」をつける

bit3　0＝意味なし、1＝正の場合も符号「＋」をつける

bit2　0＝意味なし、1＝数値の後ろに符号をつける

bit1　未使用

bit0　0＝固定小数点、1＝浮動小数点

B　小数点を含まない、小数点以前の桁数

C　小数点を含む、小数点以降の桁数

戻り値

HL　文字列の先頭アドレス

ラベル	アドレス	機　能
FOUTB	371AH	DAC+2、+3に入っている2バイト整数を2進表現の文字列に変換する
FOUTO	371EH	DAC+2、+3に入っている2バイト整数を8進表現の文字列に変換する
FOUTH	3722H	DAC+2、+3に入っている2バイト整数を16進表現の文字列に変換する

コール手順

DAC+2　2バイト整数

VALTYP　2

戻り値

HL　文字列の先頭アドレス

注　意

レジスタはいずれも保存されない。
出力ルーチンにおける文字列先頭アドレスは、通常、【FBUFFR(F7C5H〜)】内となるが、場合により先頭アドレスは前後する。
FOUTB、FOUTO、FOUTHに限らずDAC+2に入れた整数を扱うときには、【VALTYP(F663H番地)】に必ず「2」を入れなければならない。

7. 型変換

ラベル	アドレス	機　能
FRCINT	2F8AH	DACを2バイト整数にする (DAC+2、+3)
FRCSNG	2FB2H	DACを単精度実数にする
FRCDBL	303AH	DACを倍精度実数にする
FIXER	30BEH	DAC ← SGN(DAC)×INT(ABS(DAC))

注　意

実行後、【VALTYP(F663H)】にはDACの型を表す数（2、4、8）が入る。
レジスタはいずれも保存されない。

8. 整数演算

ラベル	アドレス	機能	変化するレジスタ
UMULT	314AH	DE←BC×DE	A、B、C、D、E
ISUB	3167H	HL←DE-HL	すべて
IADD	3172H	HL←DE+HL	すべて
IMULT	3193H	HL←DE×HL	すべて
IDIV	31E6H	HL←DE÷HL	すべて
IMOD	323AH	HL←DE mod HL(DE←DE/HL)	すべて

注意　ISUB、IADD、IMULT、IDIV、IMODについては、演算結果はDAC+2にも入る。

9. べき乗

ラベル	アドレス	機能	基数	指数	結果
SNGEXP	37C8H	単精度実数のべき乗を求める	DAC	ARG	DAC
DBLEXP	37D7H	倍精度実数のべき乗を求める	DAC	ARG	DAC
INTEXP	383FH	2バイト整数のべき乗を求める	DE	HL	DAC

注意　レジスタはいずれも保存されない。

6.3.3 Math-PackをMSX-DOS上のプログラムから使用する方法

Math-Packはその本体の大部分はPage 0にありますが、一部のルーチンがPage 1にあるために、MSX-DOSからコールした場合に正しい計算ができません。

それを回避するためには、以下のようにします。

リスト2.11　Math-PackをDOS環境で使用する方法

```
406fh、4666h、4eb8h、5439hにそれぞれ以下のルーチンを置く。
calbas    equ     0159h
;
;           at 406fh
;
            ld ix,406f
            jp calbas
;
;
;           at 4666h
```

```
;
        ld ix,4666
        jp calbas
;
;       at 4eb8h
;
        ld ix,4eb8
        jp calbas
;
;       at 5439h
;
        ld ix,5439h
        jp calbas
```

5597h に以下のルーチンを置く。

```
;
;       at 5597h
;
getypr:
        ld      a,(valtyp)
        cp      8
        jr      nc,double
        sub     3
        or      a
        scf
        ret
;
double:
        sub     3
        or      a
        ret
```

66a7h に以下のルーチンを置く。

```
;
;       at 66a7h
;
ppswrt:
        pop     af
        ret
```

6.4 MSX漢字ドライバ

MSX漢字ドライバは、BASICやDOSで統一的に日本語入出力を行うためのシステムソフトウエアです。

6.4.1 特長

MSX漢字ドライバには以下のような特長があります。

1. BIOSを使って文字の入出力をしているソフトウェアであれば、プログラムを変更しないで漢字の入出力ができる。
2. テキスト画面で、文字ごとに色を変えることができる。
3. JIS第二水準の漢字をサポートしている。
4. 漢字プリンタへの漢字出力をサポートしている。
5. MSX-JE(MSX日本語フロントエンドプロセッサ)がなくても、単漢字変換機能で漢字の入力ができる。
6. グラフィック画面に対し、LOCATE文やPRINT文で文字の表示ができる。

6.4.2 注意点

1. PRINT文

MSX漢字ドライバでは、グラフィック画面に対して、PRINT文を使って文字を表示することができますが、以下の点を注意しなければなりません。

1. LOCATE文で指定するカーソルポジションは、半角の文字単位です。
2. カーソルポジションを保存しているワークエリアはひとつしかないので、SET PAGE命令でアクティブページを変更しながらPRINTすると、他のページで変更したカーソルポジションの続きに文字が表示されます。
3. インターレスモード(KANJI2、KANJI3)で文字を表示する際には、SCREEN文でEVNE/ODDのインターレスモード(SCREEN ,,,,,3)に設定して、SET PAGE文で表示ページを奇数ページに、アクティブページを表示ページ-1に設定しなければなりません。

2. SCREEN文

従来は、SCREENモードを変更しただけでは、スプライト表示禁止ビット(VDPレジスタ8)やインタレースモード（VDPレジスタ9）は変化しませんでしたが、漢字モードでは両方ともクリアされます。ただし、インタレースモードはSCREEN文の第6パラメータを同時に指定した場合は、正しく設定されます。

3. ユーザーエリア

MSX-JEが実装されているシステムで、CALL KANJI命令を実行すると、MSX-JE用のワークエリアが確保されます。一度確保されたMSX-JE用のワークエリアはたとえ、CALL ANK命令を実行した場合でも解放することはできません。

そのため、アプリケーションプログラムによっては、メモリ不足のために漢字モードからは起動できないものがあるかもしれません。その場合は、一度リセットしてANKモードで起動してください。

なお、確保されるワークエリアの大きさは、フロントエンドプロセッサによって異なります。

6.4.3 拡張BIOS

漢字ドライバは、アプリケーションソフトウェアからその機能の一部を使用できるように、拡張BIOSを準備しています。MSX漢字ドライバの拡張BIOSは、エントリアドレステーブルは持たずに0FFCAH番地を呼び出します。拡張BIOSでは、現在の画面モードを返したり、画面を希望するモードに変更できます。

漢字は一文字出力のBDOSコールや、BIOSのCHPUTをコールして表示します。

1. 拡張BIOSの使用法

MSX漢字ドライバの拡張BIOSは以下の手順で使用します。

スタックポインタは0C000H番地以上を指すようにしてください。

1. Dレジスタに MSX漢字ドライバのデバイス番号である11Hを、Eレジスタに拡張BIOSファンクション番号を入れる。
2. Eレジスタにセットした拡張BIOSファンクションに必要なレジスタの設定を行う。
3. 0FFCAHをコールする。

6.4.4 拡張 BIOS ファンクション

モードの取得

機　能

現在の画面モードを数値で返します。

入　力

E　　0（モードの取得）

A　　インストール検出（下記参照）

出　力

A　　モード

0 = ANK

1 = KANJI0

2 = KANJI1

3 = KANJI2

4 = KANJI3

解　説

漢字ドライバがインストールされていない場合は[A]が不変で返ってきます。これを利用して、[A]を0FFHにして呼ぶことにより、漢字ドライバがそもそもインストールされているのか否かが判ります。単にモードが知りたいだけなら、[A]を0にして呼んでください。

モードの設定

機　能

画面モードを設定します。

入　力

E　　1（モードの設定）

A　　モード

0 = ANK

1 = KANJI0
2 = KANJI1
3 = KANJI2
4 = KANJI3

出　力

なし

解　説

スクリーンを指定されたモードに設定します。漢字ドライバがインストールされていないと無視されます。

6.5 漢字 ROM

6.5.1 漢字 ROM の I/O ポート

漢字 ROM は以下の I/O ポートに接続されています。

表 2.21　漢字 ROM と I/O ポート

アドレス	用　途	IN	OUT
第 1 水準			
0D8H	第 1 水準漢字 ROM	フォントデータ	漢字番号の下位 6 ビット
0D9H			漢字番号の上位 6 ビット
第 2 水準			
0DAH	第 2 水準漢字 ROM	フォントデータ	漢字番号の下位 6 ビット
0DBH			漢字番号の上位 6 ビット

6.5.2 フォントデータの読み出し

MSX の漢字 ROM は、読み出したい文字の漢字番号を 1 回設定した後、32 回 I/O ポートを読み出すとデータを獲得できます。内部にアドレスカウンタを持っているので、番号を変えてフォ

ントデータを読み込む必要はありません。

1. 漢字番号の計算方法

	区	漢字番号
第一水準	〜15	(区×96+点)
	16〜	(区×96+点−512)
第二水準	((区−48)×96+点)	

フォントイメージの並び方は8バイトごとに、左上、右上、左下、右下、の順です。

また、MSXの漢字ROM仕様では、フォントの形は決まっていません。フォントパターンは各メーカーによって異なっています。しかし、MSXの漢字ROM仕様では、ある特定のフォントのパターンを規定しており、そのパターンの有無をチェックすることにより、漢字ROMがシステムに実装されているかどうかの判断を行います。

JISの区点コードで規定されていない部分は統一されていません。

2. 第一水準漢字ROMのチェック

第一水準では、JISコード2140H（1区32点）の左上部分が、次のようなフォントになっています。

```
○○○○○○○○
○●○○○○○○
○○●○○○○○
○○○●○○○○
○○○○●○○○
○○○○○●○○
○○○○○○●○
○○○○○○○●
```

3. 第二水準漢字ROMのチェック

第二水準では、JISコード737EH (83区94点) の漢字（◉）の始め8バイトのデータのチェックサムが95Hになるようになっています。

例　01 02 0C 37 C0 3B 2A 2A

リスト2.12　第二水準漢字ROM実装のチェックプログラム

```
;
;
;  第二水準漢字ROMが実装されているかどうかを調べるルーチン
```

```
;
        .Z80
;
; followings are definitions of Kanji ROM port addresses
;
kan2w0  defl    0dah            ; Kanji ROM port address of lower 6bit
kan2w1  defl    0dbh            ; Kanji ROM port address of higher 6bit
kan2rd  defl    0dbh            ; Kanji ROM port address of data read

;
; このプログラムは、JIS コードの 737EH (83 区 94 点) のフォントデータのうち、
; 最初の8バイトを読んで、チェックサムを計算します。そして、その値が 95H
; であれば、Z フラグをセットしてリターンします。
;

kancod  defl    737eh
kanval  defl    95h
;
; Calculate Kanji ROM address
;
romtmp  defl    kancod-5000h    ; Subtract offset of 1st level Kanji code
romadd  defl    (high romtmp) * 96+(low romtmp)-20h
romlog  defl    romadd mod 64
romhi   defl    romadd/64

kanchk: ld      a,romlow        ; Low order byte
        out     (kan2w0),a      ; Set Kanji ROM lower 6 bit address
        ld      a,romhi         ; High order byte
        out     (kan2w1),a      ; Set Kanji ROM higher 6 bit address
        ld      c,0             ; Clear check sum accumulator
        ld      b,8             ; Loop count
chklop: in      a,(kan2rd)      ; Read the character data
        add     a,c             ; Calculate check sum
        ld      c,a             ; Update the check sum
        djnz    chklop          ; loop till all done
        cp      kanval          ; Set zero flag if Kanji ROM found
        ret

        end
```

6.5.3 デバイスイネーブル

漢字 ROM を内蔵する MSX システムは、外部スロットから漢字 ROM が挿入された場合のバスの競合を避けるために、システムソフトウェアにより内蔵 ROM のアクセスを制御しなくてはなりません。つまり、I/O アドレス F5H 番地のビット 1 に 0 が書き込まれた場合は、外部に漢字 ROM が存在することを示しているわけですから、内蔵 ROM のアクセスを禁止しなければなりません。

6.5.4 JIS から区点コードへの変換例

リスト 2.13　JIS から区点コードへの変換例

```
10 DEFINT A-Z
20 DIM P(31)
30 'INPUT JIS CODE IN HEX(4 DIGITS) THEN CONVERT TO ROM CODE
40 INPUT N$:L = VAL("&H"+RIGHT$(N$,2)):H = VAL("&H"+LEFT$(N$,2))
50 N =(H-32) * 96+L-32:IF H =>&H30 THEN N = N-512
60 L = N AND 63:H = N¥64
70 'ROM CODE OUT
80 OUT &HD8,L
90 OUT &HD9,H
100 PRINT "LOW =";L,"HIGH =";H
110 'GET PATTERN
120 FOR I = 0 TO 31
130 P(I)= INP(&HD9)
140 NEXT I
150 'PRINT OUT PATTERN
160 FOR I = 0 TO 7
170 P$= RIGHT$("0000000"+BIN$(P(I)),8):GOSUB 250
180 P$= RIGHT$("0000000"+BIN$(P(I+8)),8):GOSUB 250:PRINT
190 NEXT I
200 FOR I = 16 TO 23
210 P$= RIGHT$("0000000"+BIN$(P(I)),8):GOSUB 250
220 P$= RIGHT$("0000000"+BIN$(P(I+8)),8):GOSUB 250:PRINT
230 NEXT I
240 GOTO 40
250 FOR J = 1 TO 8
260 IF MID$(P$,J,1)="0" THEN PRINT " "; ELSE PRINT "*";
270 NEXT J
280 RETURN
```

6.5.5 シフト JIS から JIS への変換例

ファイル上で上位（変数 H)、下位（変数 L）の順番に 2 バイトづつデータが入っている場合、

リスト 2.14　シフト JIS から JIS への変換例

```
100 IF &H80<= H AND H<&HA0 THEN H = H-&H70:GOTO 130
110 IF &HE0<= H AND H<&HF0 THEN H = H-&HB0:GOTO 130
120 GOTO 160
130 H = H+H:IF L>=&H80 THEN L = L-1
140 IF L<&H9E THEN H = H-1 ELSE L = L-&H5E
150 L = L-&H1F
160 REN End conversion
```

の方法を取り、HとLを16進4桁の上位、下位に変換する。

6.5.6 サンプルプログラム

添付のフロッピーディスクに、漢字ROMをアクセスするためのサンプルプログラム(ファイル名「KANJIROM.MAC」が入っています。

7章 プログラム開発の諸注意

7.1 BIOS

7.1.1 BIOS

アプリケーションプログラムが入出力を行う時には、必ずBIOSを使って下さい。BIOSを使う目的は、ハードウェアとソフトウェアを分離し、ハードウェアが変更されてもソフトウェアをそのまま使えるようにする事です。

BIOSはROMの0000H番地から始まるジャンプテーブルを通して呼び出されます。ジャンプテーブルの飛び先、つまりBIOSの内容はハードウェアの変更や機能の拡張のために変更されますので、アプリケーションプログラムがそこを直接に（ジャンプテーブルを通さずに）参照してはいけません。MSX2の拡張ROMにも、同様のジャンプテーブルがあり、拡張機能の呼び出しに使われます。

7.1.2 拡張BIOSコールの方法

MSXでは、新たに追加されたデバイスの持つドライバールーチン（拡張BIOS）をアプリケーションソフトウェアから利用できます。その方法はデバイスによって異なりますので、詳細については各デバイスの仕様書をご覧下さい。

以下では、RS-232Cを例にとって、拡張BIOSコールの手順を説明します。

1. 8000H番地以上に拡張BIOSのエントリーアドレスを入れるためのワークエリア（64バイト）をとる。

2. HLレジスタに、1.で取ったワークエリアの先頭アドレスを入れる。
3. Bレジスタに、1.で取ったワークエリアがが存在するRAMのスロットアドレスを入れる。RAMのスロットアドレスについては、「7.3.5 RAMのあるスロット」参照。
4. Dレジスタにデバイス番号、Eレジスタに拡張BIOSのファンクション番号(0)を入れる。
5. 【EXPBIO(0FFCAH)】をコールする。
6. その結果、1.で取ったワークエリアに、以下の順番でデータが入ってくる。

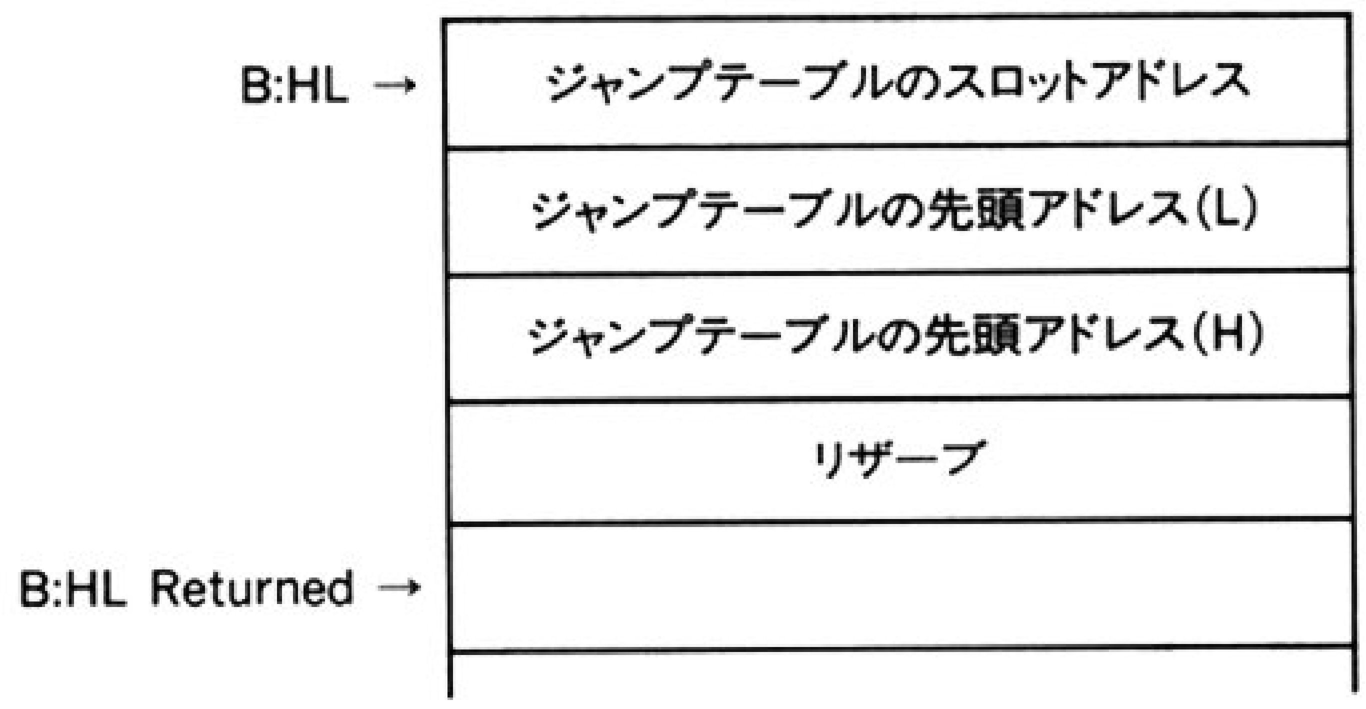


図2.33　RETURN情報の形式

ジャンプテーブルのスロットアドレスとは、拡張BIOSを持ったROMの存在するスロットアドレスを、ジャンプテーブルの先頭アドレスとは拡張BIOSへのジャンプテーブルのアドレス(EXBTBL)を指します。例えば、RS-232Cの場合、ポートをOPENする場合には、EXBTBL+6バイトをインタースロットコールします。拡張BIOSの詳細は、各デバイスの拡張BIOSコール仕様書をご覧下さい。

拡張BIOSコールを使用するデバイスには、RS-232C、MSX MODEM、MSX-AUDIO、MSX-JEなどがあります。

7.1.3 拡張BIOSのワークエリア

拡張BIOSを使うときには、スロットが切り換えられてもCPUがRAMを参照できるように、スタックをC000H番地よりも上位に置いて下さい。

同様の理由で、RS-232CなどのFCBは8000H番地よりも上位に置いて下さい。

7.2 I/O ポート

7.2.1 I/O ポート

MSX では互換性と拡張性を維持するために、ハードウェアについては基本的な仕様を決めるにとどめ、ハードウェアの違いをソフトウェアによって吸収するように定めています。したがって、市販を目的としたソフトウェアでは、その内容を問わず決して I/O ポートを直接アクセスしないで下さい。I/O ポートを直接アクセスするプログラムは各システムでの互換性を確実に失ってしまいます。ただし、VDP に関してはこの限りではありません。

7.2.2 ユーザー定義用 I/O ポート

I/O の 00H～3FH はユーザー用に解放されています。専用システムの構築などの目的で、ハードウェアを接続するときは、ここに割り当てて下さい。

40H～7FH は周辺機器用にリザーブされています。市販を目的としたハードウェアはなるべくメモリマップド I/O 方式で本体と接続して下さい。スピード等の問題で、どうしても I/O ポートに接続することを希望する場合は、あらかじめ株式会社アスキー　MSX 推進部(連絡先は巻末を参照) までご連絡下さい。当社で I/O の割り当て調整を行います。

7.3 スロット

7.3.1 拡張スロット

過去に、

1. 拡張スロットに差し込まれた場合
2. RAM が拡張スロットにある場合

などに、動作しないソフトウェアがいくつかありました。MSX2 の多くの機種では本体内部で拡

張スロットを使っていますので、拡張スロットでの動作不良は致命的な問題になります。ソフトウェアを発売するまえに、そのソフトウェアが拡張スロットに入れられた場合と、RAM が拡張スロットにある場合について、必ず動作試験を行って下さい。

特に、FFFFH 番地には拡張スロットレジスタが置かれますので、そこを RAM のつもりで参照してはいけません。拡張スロットで暴走するソフトウェアの多くが、

```
LD      SP,0
```

を行っていました。このようなソフトウェアを作らないようにご注意下さい。

7.3.2 プログラムのスロットアドレス

プログラムが1ページと2ページ (4000H 番地から BFFFH 番地) までの 32K バイトの ROM に入っている場合には、1ページの ROM の INIT ルーチン(データブックのスロット管理メカニズム参照)が呼び出された時に、2ページがその ROM の続きではなく、別のスロットになっています。1ページの ROM の内部で自分がいるスロットのスロットアドレスを調べ、2ページをそのスロットに切り換える必要があります。

添付のフロッピーディスクにサンプルプログラムが入っています。ファイル名は「WHERE AMI.MAC」です。

他のプログラムから参照されるような場合にも、同じ方法で求めた自分のスロットアドレスを利用して下さい。

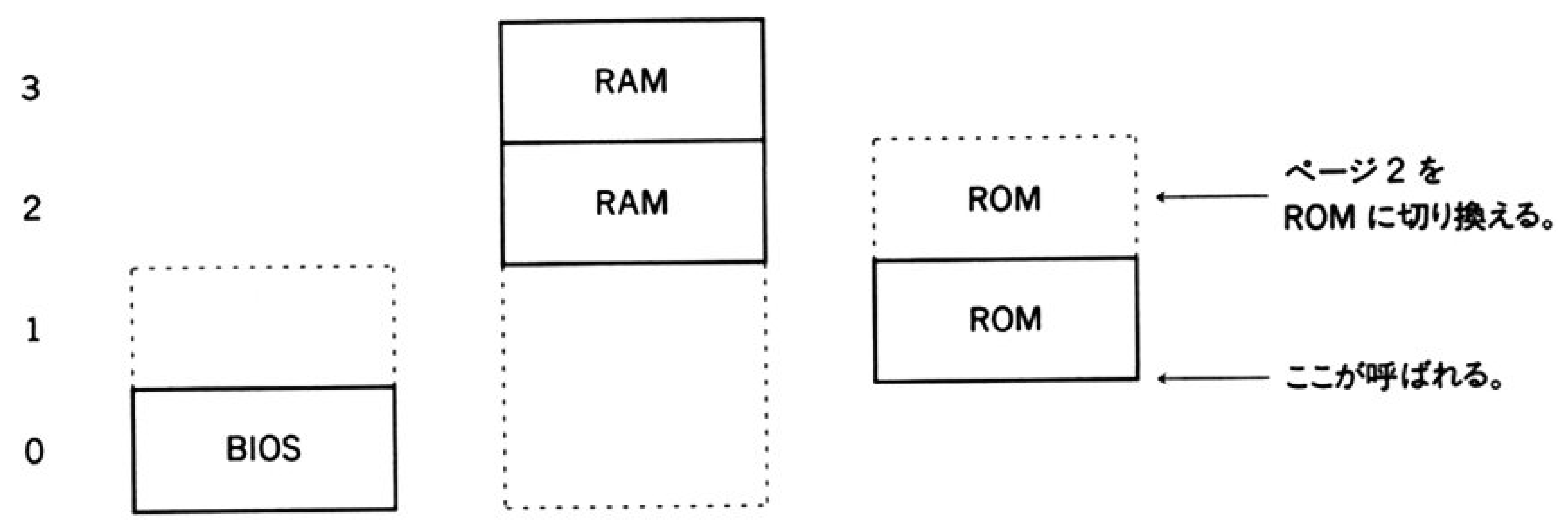


図 2.34 INIT ルーチン呼び出し時のスロット状況例

7.3.3 BASICのROMのスロットアドレス

従来は、「BASICのROMは基本スロットの0、または基本スロットの0を拡張した0-0に置く。」という仕様がありました。しかし、本体外にV9938とROMを増設してMSX2の機能を持たせる場合には、BASICのROMが他のスロットにあります。また、MSX2の拡張ROMが入っているスロットは、機種によって異なります。

ゆえに、下記のシステムワークエリアの内容を参照して、BASICのROMのスロットアドレスを調べて下さい。

DOSからBIOSを呼ぶ場合にこの方法を使えます（「5部 スロットとカートリッジ」参照）。

表2.22 BASICのROMのスロットアドレス

アドレス	ラベル	内容
FCC1H	EXPTBL	MAIN-ROMのスロットアドレス
FAF8H	EXBRSA	SUB-ROMのスロットアドレス

注 意　BASIC 1.0では、EXBRSAの内容は00Hです。

7.3.4 スロット切り替え時の注意

インタースロットコールを行うと、IX、IY および裏レジスタの内容が破壊され、リターン時に割り込みが禁止されています。

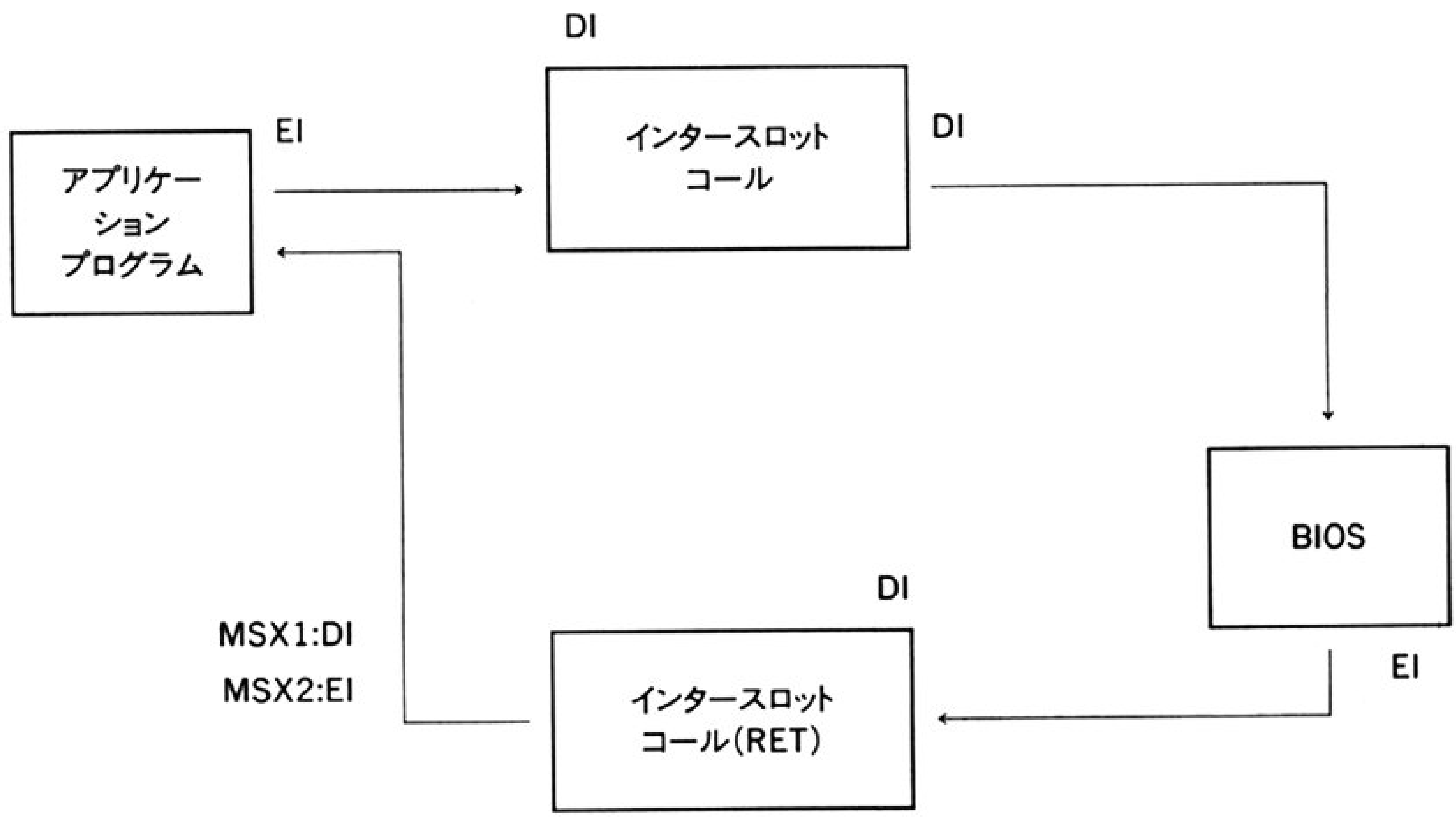


図 2.35 インタースロットコール時の割り込み

Z80 の特性により、MSX2 でもごく希なケースとして、DI 状態でリターンすることがあります。割り込み状態が EI の必要があるプログラムでは、インタースロットコール直後に EI することをお勧めします。

7.3.5 RAM のあるスロット

フロッピーディスクドライブを実装しているシステムでは、各ページの RAM のあるスロットを以下のワークエリアに保存しています。フロッピーディスクドライブが実装された MSX で、RAM の実装されたスロットアドレスを知りたい場合には、必ずこのワークエリアを参照して下さい。

例えば、ページ 1 の RAM をデータエリアとして使用するためにスロットを切り換えるといった場合に、このワークエリアの値を参照します。

表2.23 RAMのスロットアドレスを保存したワークエリア

RAMのスロット	ラベル	アドレス
ページ0	RAMAD0	0F341H
ページ1	RAMAD1	0F342H
ページ2	RAMAD2	0F343H
ページ3	RAMAD3	0F344H

各ビットは以下の意味を持ちます。

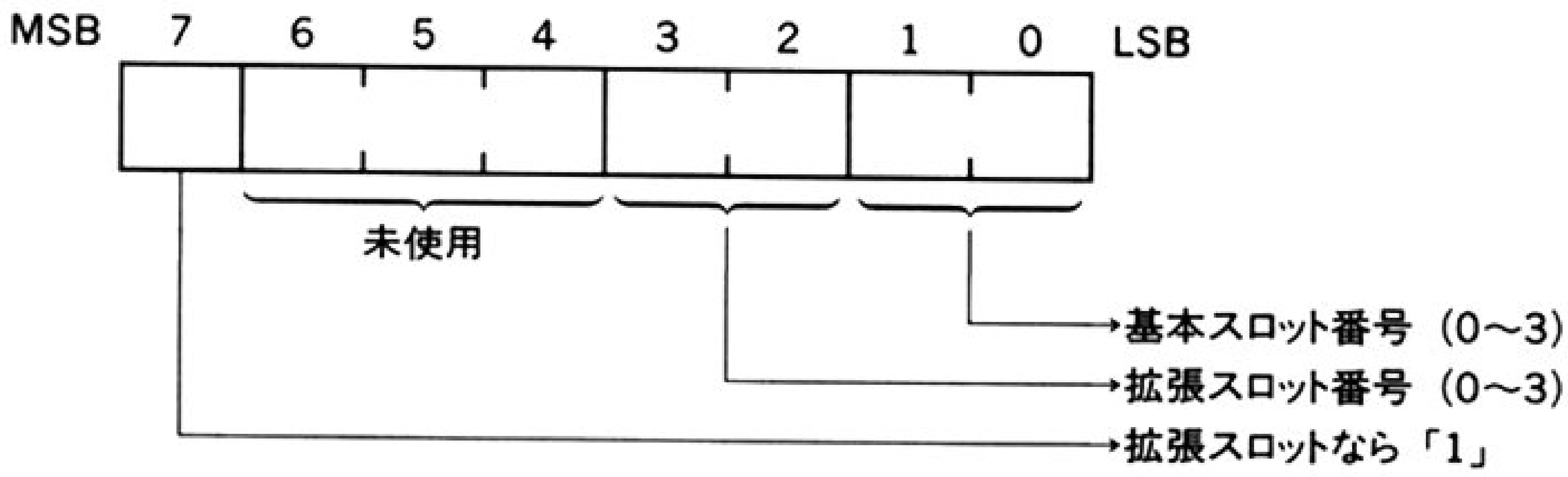


図2.36 RAMAD0～3のビットの意味

フロッピーディスクドライブを実装しないシステムの場合は、RAMのあるスロットを自分で捜さなければなりません。添付のフロッピーディスクに、「RAMSRCH.MAC」という名前のサンプルプログラムが入っています。

このプログラムはBASICプログラムからDEFUSR命令で呼ばれるようになっています。フロッピーディスクドライブを実装した機種では、上記のワークエリア【RAMAD0(0F341H)】～【RAMAD3(0F344H)】を参照し、このサンプルプログラムによるRAMサーチは行わないで下さい。さもなければ、システムに矛盾をきたし、暴走の原因となる場合があります。

7.4 ワークエリア

7.4.1 BASICのワークエリア

メインRAMのF380H番地からFFFEH番地までは、BIOSとBASICインタープリタのワークエリアですから、不用意に使わないで下さい。ワークエリア内の使われていない場所は、将来

の機能拡張のために予約されています。

7.4.2 ユーザーエリア

BASIC、Disk BASIC の環境で、ユーザーの使用できる RAM エリアは 8000H 番地～0FC4AH と 0FC4BH の内容が示すアドレスです。0FC4AH のラベル名は HIMEM です。

MSX-DOS では、CP/M と同じく、100H 番地～6、7H 番地の内容が示すアドレスまでが TPA となります。

アプリケーションプログラムは必ず HIMEM（DOS の場合は 6、7H 番地）を確かめて、最悪の場合でも暴走しないように対処する必要があります。対処の方法としては、

1. ワークエリアをリロケータブルにする。
2. ワークエリアを BOTTOM から確保する。
3. ドライブの数を減らすように指示して停止する。

などがあります。

7.4.3 RAM とスタックポインタの初期化

電源投入時の RAM の内容は不定で、システムのワークエリア以外の領域は初期化されません。自分のワークエリアを初期化する事は、アプリケーションプログラムの責任です。

RAM の内容が 00H である事を期待しているために、特定の本体との組み合わせでしか動かないソフトウェアがあり、問題になっています。

また、ROM カートリッジの INIT ルーチンが呼ばれた時のスタックポインタの値は不定で、ディスクインターフェイスが先に初期化されたような場合にはそうでない場合よりもスタックポインタの値が小さくなっています。このため、スタックポインタを初期化しないプロガラムが暴走するという問題が起きました。INIT ルーチンで起動し、そのまま走り続けるプログラムは、必ずスタックポインタを初期化して下さい。

7.4.4 デバイスドライバなどのワークエリア

デバイスドライバや BASIC から呼ばれるサブルーチンのように、他のプログラムと同時にメモリに存在するプログラムは、ワークエリアの確保に特に注意を要します。

カートリッジのINITルーチンがHIMEM(またはBOTTOM)を書き換え、古いHIMEMと新しいHIMEMの間を自分のワークエリアとして予約します。そのワークエリアのアドレスをSLTWRKというスロット別に割り当てられた2バイトの領域に記録します。

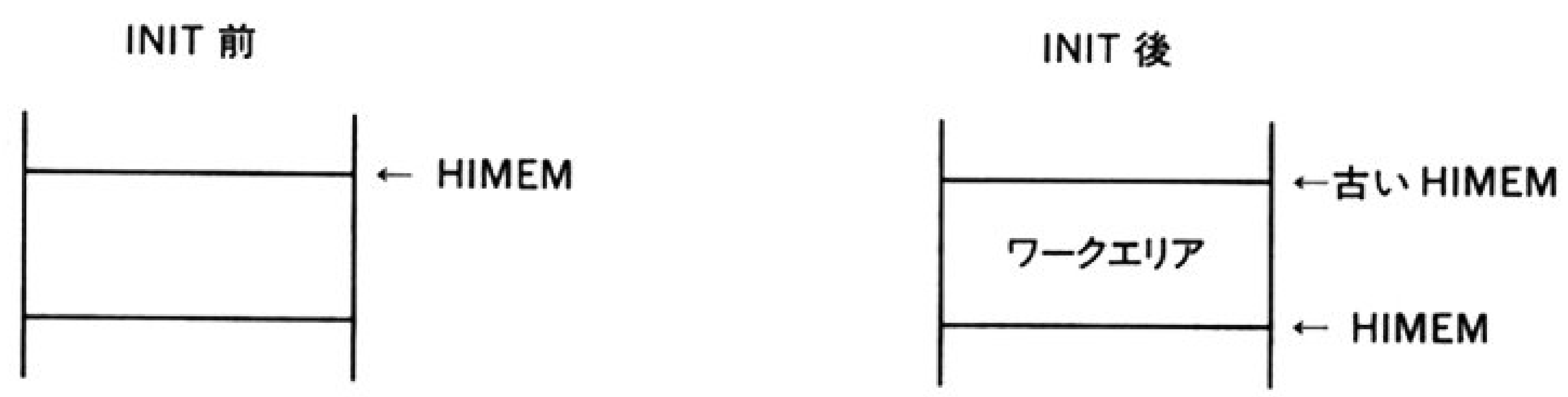


図2.37 デバイスドライバのワークエリア

7.4.5 フック

例えば、RS-232Cカートリッジが割り込み処理のためにフックを書き替えたとします。次のカートリッジが同じフックを書き替えると、RS-232Cカートリッジは割り込みフックを使えなくなります。

このような事を防ぐためにフックを書き替える場合には、以前のフックの内容(例では、RS-232Cカートリッジの割り込み処理ルーチンへのインタースロットコール命令)を記憶します。そして、自分がフックで呼ばれたら以前のフックの内容を呼び出し、そのフックを使おうとしている全てのカートリッジに制御が渡るようにして下さい。

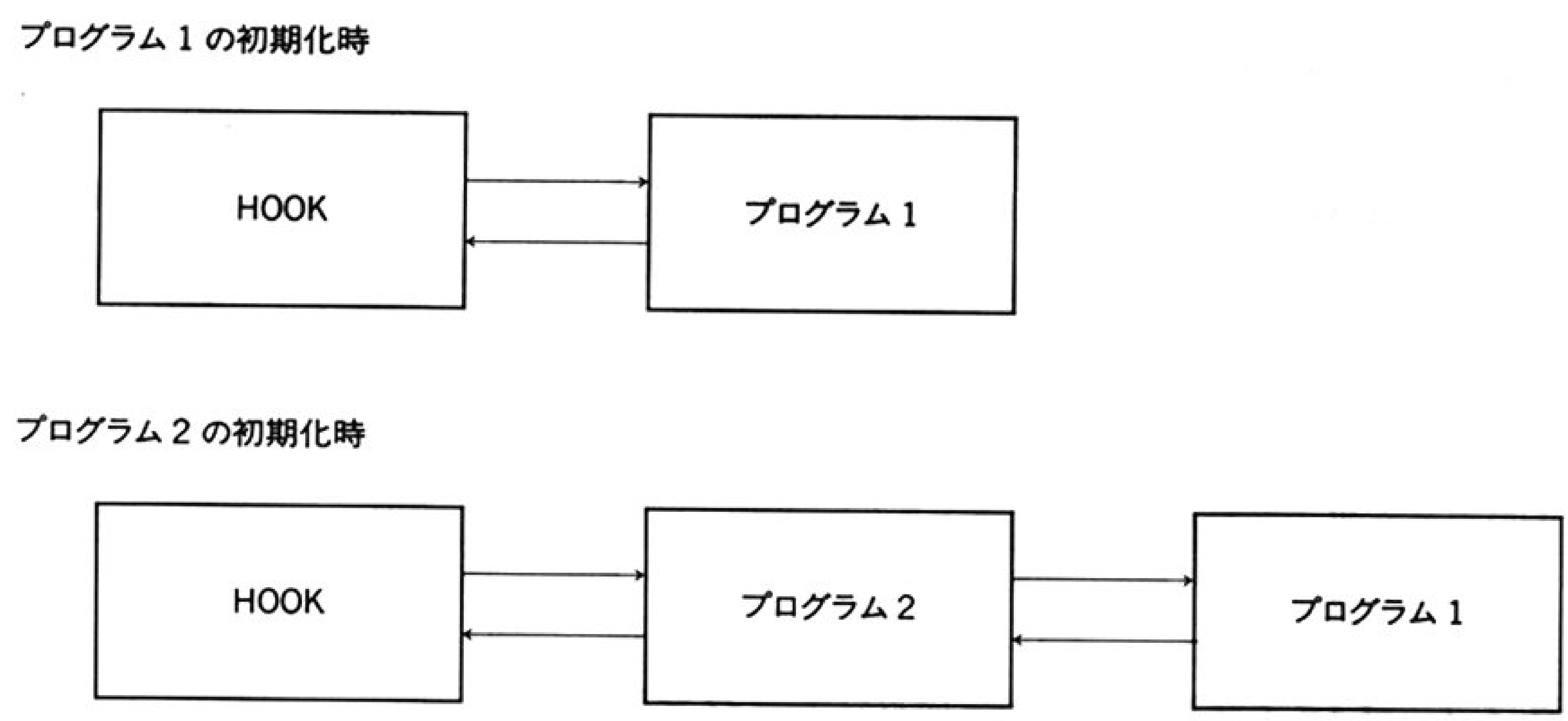


図2.38 フック

7.5 VDP

7.5.1 VDP の I/O アドレス

MAIN ROM の 6 番地と 7 番地には VDP のリードアドレスとライトアドレスが書き込まれています。I/0 ポートを使って VDP を直接アクセスする場合は、ROM の内容から I/O ポートのアドレスを調べ、その I/O アドレスをアクセスして下さい。TMS9918 を使っている機種(MSX1)では VDP が 98H 番地に置かれていましたので、その番地を直接参照しても実害はありませんでしたが、V9938 を外付けして MSX2 に拡張したハードウェア（MSX2 アダプター）の場合には VDP のアドレスが変わりますので、必ず以上の事を守って下さい。

表 2.24 VDP の I/O アドレス

VDP のポート	I/O アドレス
データリード	MAIN ROM の 6 番地の内容
データライト	MAIN ROM の 7 番地の内容
ステータスリード	データリードアドレス+1
コマンドライト	データライトアドレス+1
パレットライト	データライトアドレス+2
インダイレクトアクセス	データライトアドレス+3

7.5.2 レジスタの保存

VDP のいくつかのレジスタは書き込み専用であり、直接には現在のレジスタの内容を知る事ができません。そこで、システムワークエリアの【RG0SAV(0F3DFH)】から【RG7SAV(0F3E6H)】と【RG8SAV(0FFE7H)】から【RG23SA(0FFF6H)】(RG8SAV 以降は V9938 使用機種のみ)に、レジスタの内容を保存してあります。BIOS 通さずに VDP のレジスタを書き替える場合には、これらのワークエリアの内容も書き替えて下さい。特に、機械語プログラムから BASIC や DOS に制御が戻った時にレジスタの内容とワークエリアの内容が異なると、画面表示が正しく行われません。

また、VDP のレジスタの一部のビットを書き替える場合には、ワークエリアからレジスタの内容を読んで、必要に応じてビットを変え、そのデータをレジスタとワークエリアの両方に書き込

んで下さい。

7.5.3 VRAMの初期化

アプリケーションプログラムが動き始める時のVRAMの内容は不定です。アプリケーションプログラムは、必ずVRAMを初期化して下さい。

VRAMを初期化していないために、ハードウェアの特性でVRAMの内容が0になっている機種でのみ画面が正しく表示されるというソフトウェアがありました。

7.5.4 VRAMの容量

FAFCH番地のビット2とビット1の内容で、VRAMの容量が判ります。

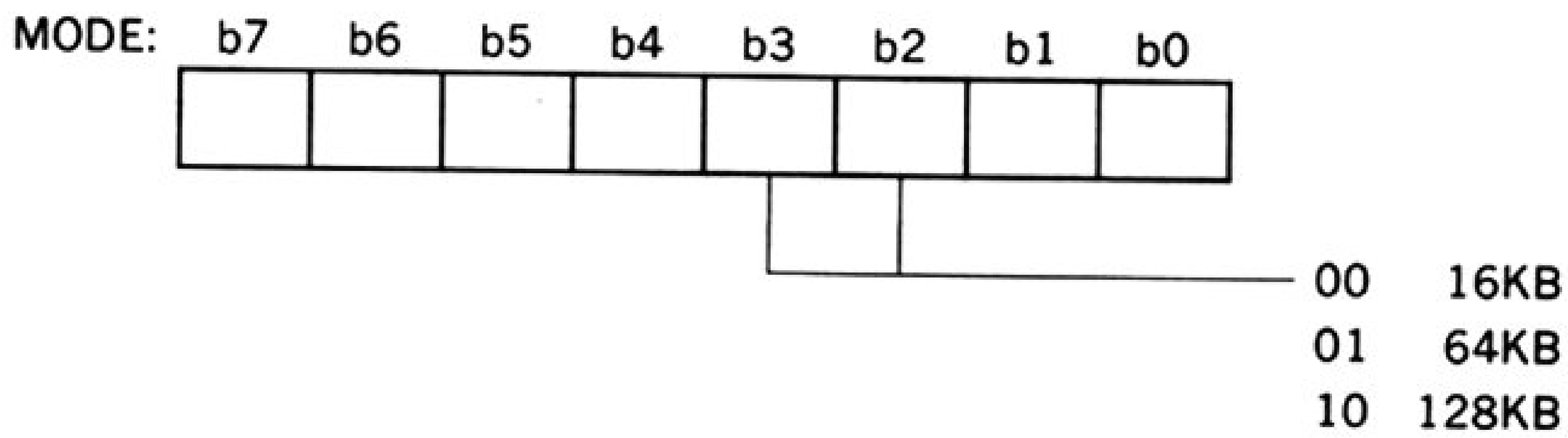


図2.39 【MODE(0FAFCH)】のビット

7.6 キー入力

7.6.1 キー入力の方法

BIOSによるキー入力に2つの方法があります。目的に応じて使い分けて下さい。特に、ヨーロッパ用などの異なる種類のキーボードに対応するソフトウェアを作る場合に注意が必要です。

1. SNSMATルーチンを使って、リアルタイムにキーの物理的な状態を得ます。
2. CHSNSとCHGETを使い、バッファリングされた文字コードを得ます。また、KILBUFルーチンでキーバッファの内容を捨てることができます。この方法では、

VDPによるタイマ割り込み時にキースキャンが行われますので、割り込みが許可されている必要があります。

7.6.2 キークリック

システムワークエリアの【CLIKSW(0F3DBH)】という1バイトの内容が0でなければキー入力時にクリック音が発生し、0であればクリック音が発生しません。

7.7 プリンタ

7.7.1 MSX用でないプリンタへの対応

【NTMSXP(0F416H)】というワークエリアの内容が0でなければ、MSX用でないプリンタのためにBIOSが文字を変換します。その場合、ひらがながカタカナに変換され(日本版のみ)、グラフィック文字が空白に変換されます。

7.7.2 RAWモード印字

【RAWPRT(0F418H)】の内容が0でなければ、RAWモージ印字となり、タブ(09H)の空白への変換と前項の文字の変換が行われません。ビットイメージプリントを行う時には、必ずRAWモードを設定して下さい。

ただし、RAWモード印字はBIOSのOUTDO(0018H/MAIN)を使用しているプログラムのみ有効で、LPTOUT(00A5H/MAIN)では無効です。したがって、BASICのPRINT命令などではRAWモード印字されますが、MSX-DOSのプリンタ出力(ファンクションコール05H)などでは何も起こりません。

MSX2+やMSX-DOS2の漢字BASICでプリンタにエスケープシーケンスなどを送る場合も、RAWモード印字にして下さい。

7.8 BASICの種類

7.8.1 BASICのバージョン番号

アプリケーションプログラムがBASICのバージョン番号を知るためには、下記の方法を使います。

MAIN ROMの2DH番地にBASICのバージョン番号が入っています。MSX BASIC 1.0では2DH番地の内容が00H、MSX BASIC 2.0では01H、MSX BASIC 3.0では02Hです。

7.8.2 インターナショナルMSX

MSXは世界各国で使われており、その種類は主なものだけで10を越えています。下記の項目が、国によって大きく異なります。

1. キーボード配列
2. 文字セット
3. PRINT USINGの書式
4. タイマ割り込み周波数

IDバイトというROM内の情報を読むと本体の種類が分かり、各国のMSXに対応できます。

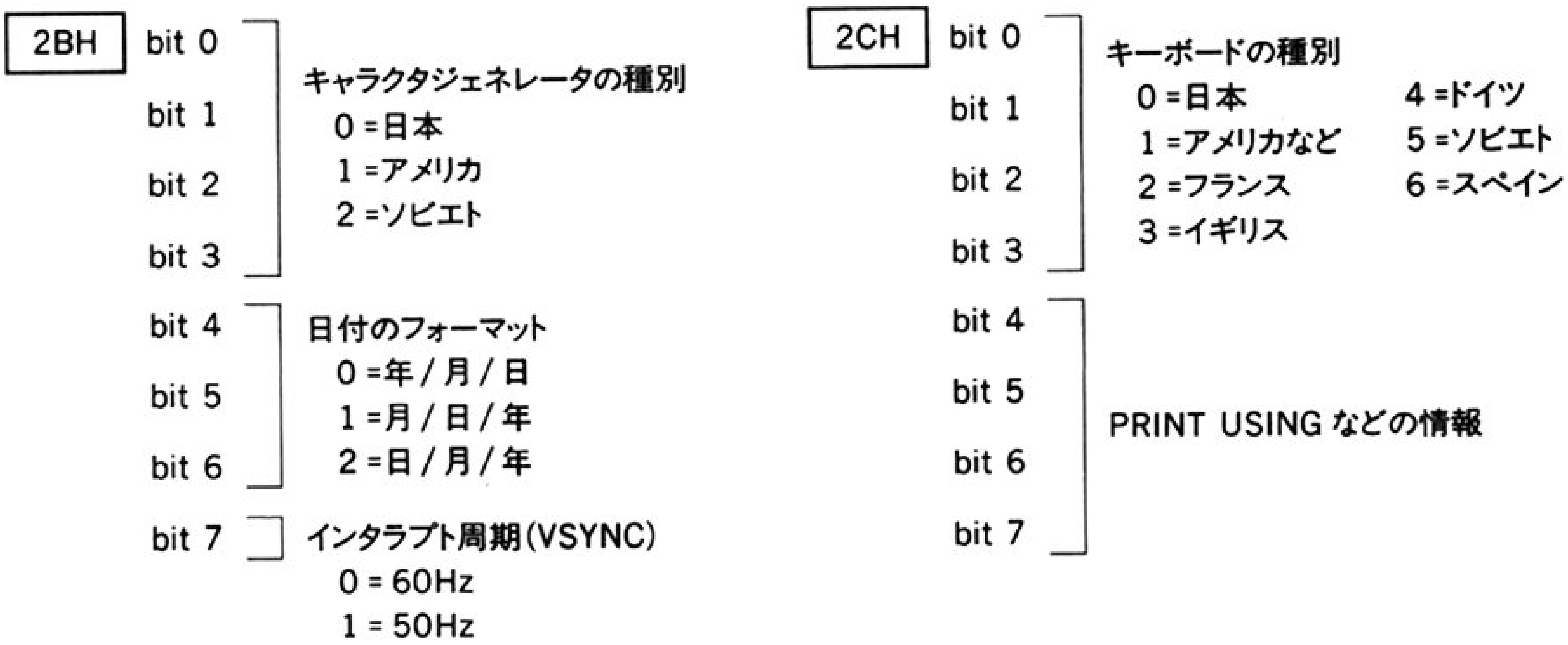


図 2.40　ID バイトの内容

表 2.25　国別の MSX 仕様

国名	TV セット	日付フォーマット	初期画面モード	PRINT USING		
				文字列の長さ指定	文字の置き換え	通貨シンボル
日本	NTSC (60Hz)	年/月/日	SCREEN 1	%	@	¥(エン)
イギリス	PAL (50Hz)	日/月/年	SCREEN 0	%	&	£(ポンド)
インターナショナル	PAL (50Hz)	月/日/年	SCREEN 0	%	&	$(ドル)
アメリカ	NTSC (60Hz)	月/日/年	SCREEN 0	%	&	$(ドル)
フランス	SECAM (50Hz)	日/月/年	SCREEN 0	%	&	$(ドル)
ドイツ	PAL (50Hz)	日/月/年	SCREEN 0	%	&	$(ドル)
ソビエト	NTSC (60Hz)	月/日/年	SCREEN 0	%	&	$(ドル)
スペイン	PAL (50Hz)	月/日/年	SCREEN 0	%	&	$(ドル)

7.9 ディスク

フロッピーディスクが使用するワークエリアの情報は、原則的には公開していません。この章で公開しているアドレスと使用法以外の目的で、ワークエリアを参照したり、書き換えたりすると動作は保証できませんので、十分ご注意下さい。また、この章で紹介しているワークエリア以外は、将来のバージョンアップの際に、アドレスや使用法が変わる可能性があります。

7.9.1 ディスクを使わない方法

リセット（または電源投入）してからビープ音が鳴るまでシフトキーを押していると、ディスクインターフェイスが無視され、ROM BASICが起動します。ディスク内蔵機でディスクとワークエリアが衝突するソフトウェアを動かす場合には、この方法を使って下さい。

同様に、コントロールキーを押していると、2ドライブシミュレーション機能(1ドライブのシステムでソフト的に2ドライブをシミュレートする機能）が無効になり、ディスクワークエリアが少し小さくなります。

アプリケーションプログラムから見てドライブが2台接続されているシステムで、それが2ドライブシミュレーション機能なのか、それとも物理的にディスクが接続されているかを知ることはできません。また、ソフトウェアから2ドライブシミュレーション機能を無効にすることも不可能です。

7.9.2 ドライブの有無と数を知る方法

【H.PHYDIO(0FFA7H)】というフックの内容を読んで下さい。そこにRET命令が書かれていれば、ディスクは接続されていません。RETでなければ、ディスクが接続されています。

ディスクが存在する場合は、DOSのシステムコール（BDOSコール）の18H（ログインベクトルの獲得)を使用して下さい。DOSのシステムコールはDisk BASICからも利用できます。詳しくは「第3部 4章 システムコール」をご覧下さい。

7.9.3 エラー処理

MSX-DOSには、ディスク入出力のエラーを検出して適切な回復手続きを行う機能が用意されています。

1. MSX-DOSの標準エラー処理手順

システムコールのディスク入出力でエラーが発生すると、エラー処理ルーチンが呼ばれます。標準処理手順では、エラーメッセージと「Abort Retry Ignore?」というプロンプトが表示されて、キー入力を待ちます。ここで、A、R、Iを入力しますが、これらはそれぞれ以下のような意味があります。

A　　システムのウォームブートが起こり、DOSのコマンド入力状態になります。

R　　処理を再び試みます。

I　　システムコールをキャンセルして、アプリケーションには正常終了を返します。

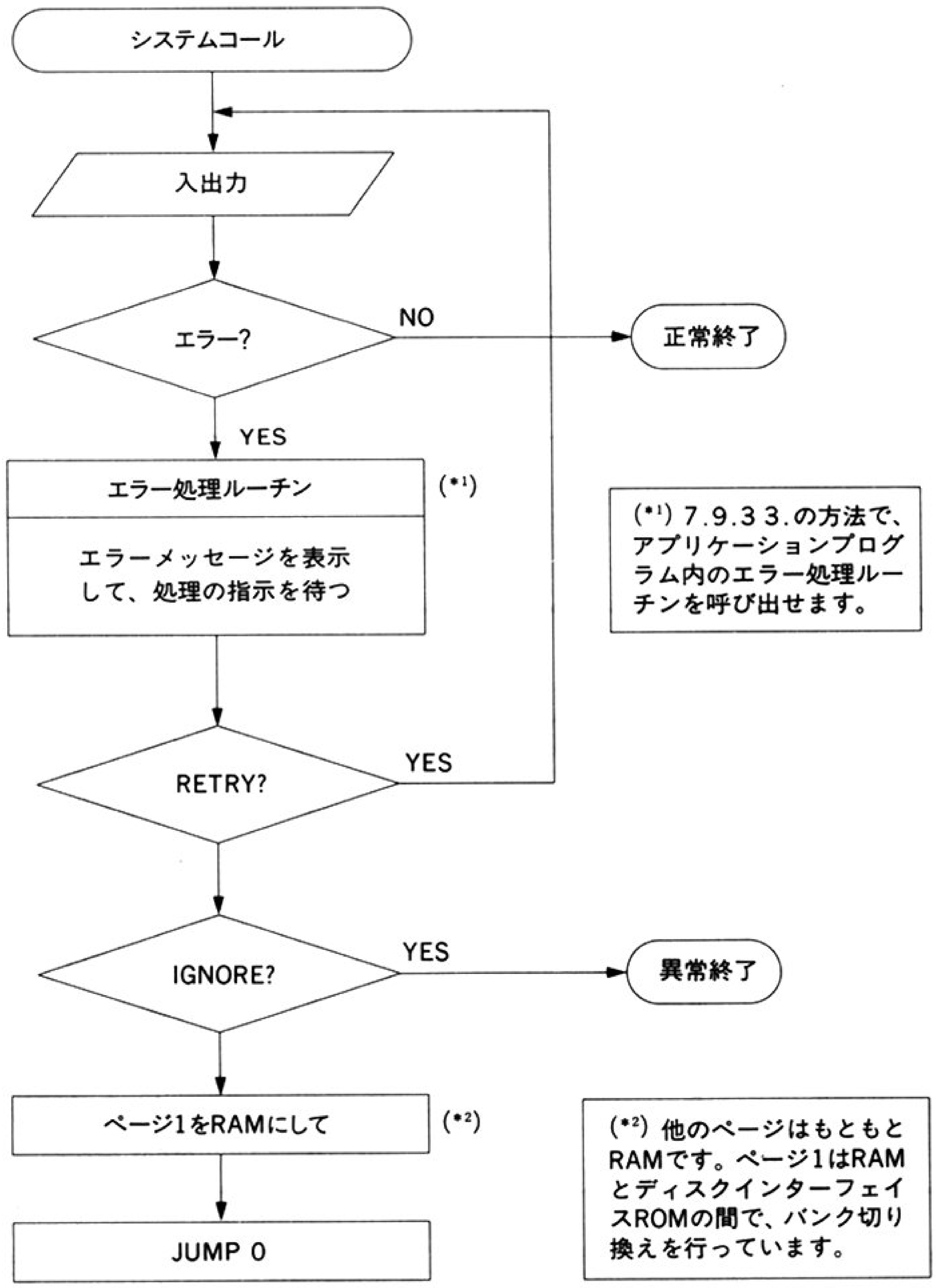


図2.41　MSX-DOSの標準エラー処理手順

2. Disk BASICの標準エラー処理

Disk　BASICの環境でディスクエラーが起こった時には、エラーメッセージを表示して、BASICの入力待ち状態になります。

3. アプリケーションプログラムがエラーを処理する方法

【DISKVE(0F323H)】は、ディスクエラー処理ルーチンへのポインタへのポインタ（エラー処理ルーチンの開始アドレスを記憶してあるアドレス)です。そこを書き替えることにより、エラー発生時にアプリケーションプログラム内のエラー処理ルーチンを呼び出すことができます。
図2.41の *1は、エラー処理ルーチンの入口を示します。ここで、

A レジスタにはエラーを起こしたドライブの番号（A = 0、B = 1...H = 7)

C レジスタにはエラーの種類

が入っています。C レジスタの各ビットの意味は以下の通りです。例えば、書込み時に、「NOT READY」エラーが起こると、C レジスタの内容は 03H になります。

表 2.26 エラーの種類

bit0(LSB)	エラー時の状態
0	読み込み時
1	書き込み時

(MSB)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	エラーの種類
1	0	0	0	0	0	0	bad FAT
0	0	0	0	0	0	0	write protect
0	0	0	0	0	0	1	not ready
0	0	0	0	0	1	0	CRC error
0	0	0	0	0	1	1	seek error
0	0	0	0	1	0	0	record not found
0	0	0	0	1	0	1	write fault
0	0	0	0	1	1	0	other error

図2.41の *2は、エラー処理ルーチンの出口を示します。ここから DOS の標準エラー処理に戻る場合には、C レジスターの内容で処理を指定し、リターンします。ここで、AF、B、DE、HL レジスターには何が入っていても構いません。

C レジスタ	事後処理
2	abort（処理を中断)
1	retry（入出力を再度試みる)
0	ignore（エラーを無視して先へ進む)

以下の場合には、エラー処理ルーチンは必ず2（abort）または1（retry）を返すようにして下さい。さもなければ、ディスクが壊れることがあります。

1. エラーコードのビット7が1（bad FAT）だった場合
2. エラーコードが1010（Unsupported media type）だった場合

Disk BASICの環境下では、エラーが発生したときには、Disk ROMがイネーブルされており、スロット切り替えは行われません。したがって、ページ1にあるディスクエラー処理ルーチンは制御を受け取ることができません。しかし、MSX-DOSの環境では、ページ1のRAMがイネーブルされてから、コントロールが渡ってきます。

DOSから帰ってくるエラーは「directory entry not found」や「no more free sectors to write」のようなロジカルエラーです。そして、ディスクエラーハンドラが返すのはハードウェアのエラーです。この2つは異なります。

MSX-DOS上で動作するアプリケーションプログラムが特別なエラー処理を行いたい場合は、下記のようにします。

1. DISKVEの内容をセーブする。
2. DISKVEにエラー処理ルーチンへのポインタへのポインタを設定する。
3. 01H番地と02H番地の内容をセーブする。
4. 1.、3.でセーブした値をRESTOREするルーチンへのエントリを01H番地、02H番地に書く。1.、3.の値をRESTOREするルーチンとは、
   a. DISKVEを元の値に戻す
   b. 01H番地と02H番地を元の値に戻す
   c. 0H番地へジャンプする
   を実行するもの。
5. システムコールの実行前に、スタックポインタを保存する。
6. システムコールを行う。
7. エラー処理ルーチンが呼び出されたら、スタックポインタを元に戻し、適当な処理を続ける。

前述の【DISKVE】の内容は、システムのコールドスタート時、およびDOSとBASICの切り換え時に初期化されます。このため、あるプログラムが【DISKVE】の内容を書き換えたまま終了し、DOSのコマンドや次のプログラムがエラーを起こすと暴走します。

ゆえに、DISKVEを書き替えたプログラムはそのままコマンドレベルに戻らずに、01H番地と02H番地の内容（WOOM BOOT）を書き替えて、CTRL＋Cでプログラムが止まらないようにして下さい。

7.9.4 BDOS コール

MSX Disk BASIC の環境下でも、DOS ファンクションコールが可能です。F37DH が Disk BASIC の場合でのファンクションコールエントリで、MSX-DOS の場合の5番地コールに相当する機能を提供します。詳しくは「第3部 4章 システムコール」を参照して下さい。

7.9.5 FDC

MSX では FDC の種類や割り当てられるアドレスは定められていません。したがって、いかなる理由があろうとも、アプリケーションプログラムは直接 FDC をアクセスしてはいけません。ディスクのアクセスには必ずファンクションコール（BDOS コール）を使用してください。

7.9.6 DISK ROM が入っているスロットを知る方法

ワークエリアの FB21H から FB28H までのドライブテーブルという領域に、ディスクインターフェイスに関する情報が記録されています。各カートリッジに2バイトが割り当てられており、その内容は以下のようになっています。カートリッジが3個以下であれば空いているテーブルに 00H が入ります。

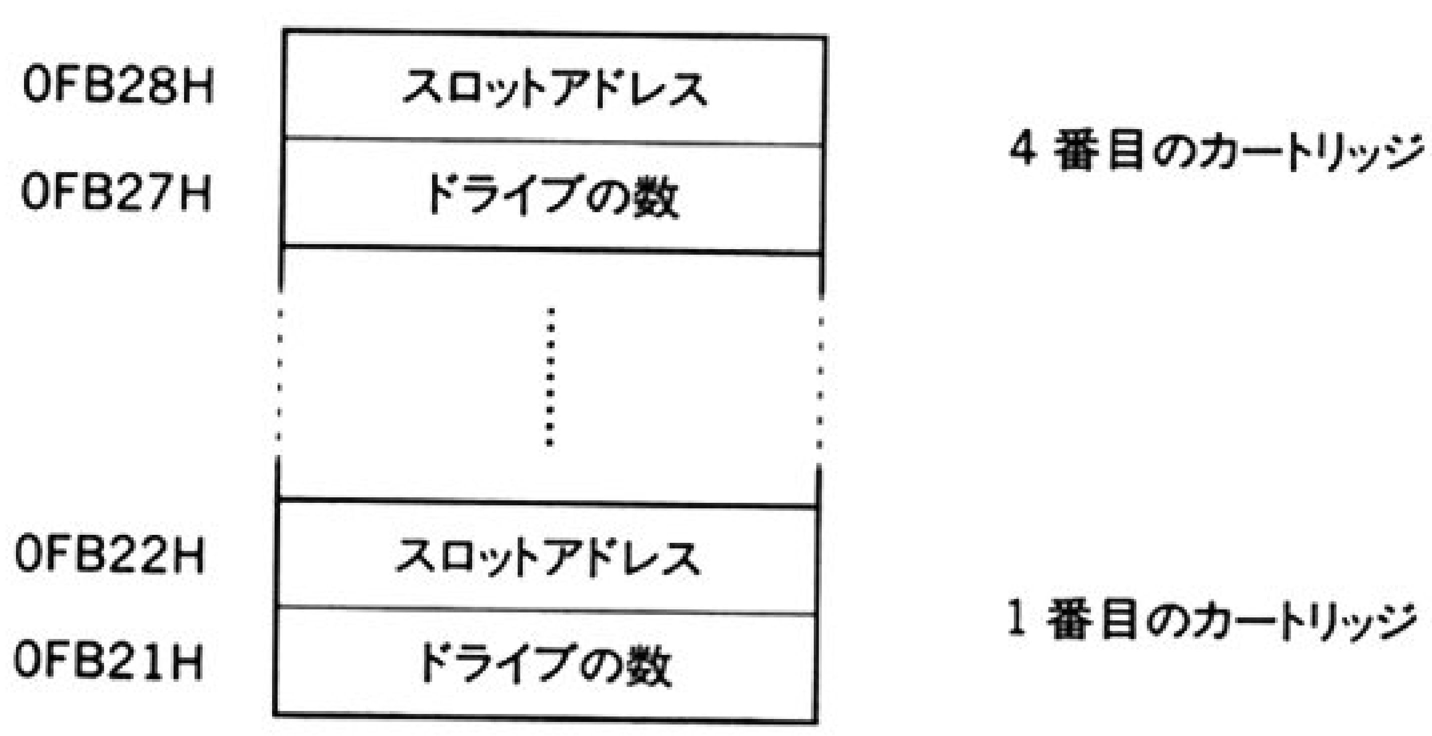


図 2.42　DISK ROM の情報

7.9.7 FDDのモーターを止める方法

以下の方法で、アプリケーションプログラムがFDDのモーターを止めることができます。ただし、ディスクドライブによっては、モーターOFFのエントリを持っていないものがあるので、そのエントリがあるかどうかを事前にチェックしてください。また、ディスクドライブによっては、何もしないでリターンするものもあります。

なお、このルーチンはアスキーから発売しているハードディスクインターフェイスで実行すると、シャットダウン(ハードディスクのヘッドを安全地帯へ退避させること)の効果があります。ユーザープログラムがシャットダウンを行うときは、ユーザープログラムから、@_SHUTDOWNをコールして下さい。

リスト2.15 FDDのモーターOFF

```
;
;       moter off Subroutine
;
;
        .z80
        entry           @_SHUTDOWN
;
rdslt           equ     000ch
calslt          equ     001ch
motor_off_entry equu    4029h
master_slot     equ     0f348h
;
@_SHUTDOWN:
        ld      a,(master_slot)         ; motor off entry present?
        ld      hl,motor_off_entry
        call    rdslt
        and     a
        ret     z                       ; no, no way....
        ld      iy,(master_slot-1)      ; we have it! call it now
        ld      ix,motor_off_entry
        jp      calslt
;
; 注意
;       マスターカートリッジの'motor_off_entry'はシステムにつながっている
;       すべてのディスクドライブを調べ、モーターを止めます。
;       しかし、旧版のマスターカートリッジはモーターを止めることができません。
;       この場合は、モーターを止める手だてはありません。
;       たとえ、マスターカートリッジがその機能を持っていても、
;       スレーブカートリッジがその機能を持っていなければ、
;       スレーブカートリッジが管理するディスクドライブのモーターは止まりません。
;
        end
```

Cコンパイラからは、_shutdown(); でコールできます。
Cコンパイラのテストプログラムです。

```
#include          <stdio.h>
VOID    _shutdown();
main()
{
        _shutdown();
}
```

7.9.8 マスターカートリッジのスロットアドレス

ディスク機能の大部分を受け持つディスクカートリッジは、「マスターカートリッジ」と呼びます。マスターカートリッジのスロットアドレスは、【MASTER_SLOT(0F348H)】に保存されています。

7.9.9 ディスクの交換

2ドライブシミュレーション機能が働いているときは、別のドライブにアクセスすると、

```
Insert disk for drive x:
and strike a key when ready
```

というメッセージが表示されます。そのため、グラフィックを使ったアプリケーションソフトウェアでは不都合が生じます。

アプリケーションソフトウェアが、自分でディスクの交換メッセージを出すためには、【H.PROMPT(0F24FH, 3)】をアプリケーションプログラム側のディスク交換ルーチンへのジャンプ命令に書き換えます。

【H.PROMPT(0F24FH, 3)】がコールされたときは、Aレジスタにドライブ名（A、B・・・）がキャラクタで入っています。また、スタックトップが標準のプロンプトルーチンへのリターンアドレスで、スタックトップ+2がシステムへのリターンアドレスになっています。したがって、標準プロンプトルーチンをバイパスするためには、リターンする前にスタックを1レベル捨てなければなりません。なお、レジスタを保存する必要はありません。

注 意	交換ルーチンはページ1には置けません。 交換ルーチンからは、アプリケーションプログラムにジャンプしてはいけません。必ずリターンして下さい。

7.10 便利な機能

7.10.1 エスケープシーケンス

MSX には別表のようなエスケープシーケンス機能があります。この機能は BASIC の PRINT 文、BIOS および BDOS コールのコンソール出力で利用できます。シーケンスの機能は、DEC 製 VT52 ターミナルおよびヒースキット製 H19 ターミナルのサブセットになっています。

7.10.2 BASIC に戻る方法

BASIC ROM のスロットをイネーブルした後、MAIN ROM の 409BH 番地へジャンプして下さい。BASIC のワークエリアが壊されていなければ、BASIC のコマンド入力の状態になり、OK (MSX2 でプロンプトの設定を行っていればその文字列) が表示されます。ジャンプする時のレジスターやスタックの内容は一切無視されます。

また、内部ルーチンの NEWSTT (「4.4 BASIC の内部ルーチン」参照) で次の命令を実行しても、BASIC に戻ることができます。

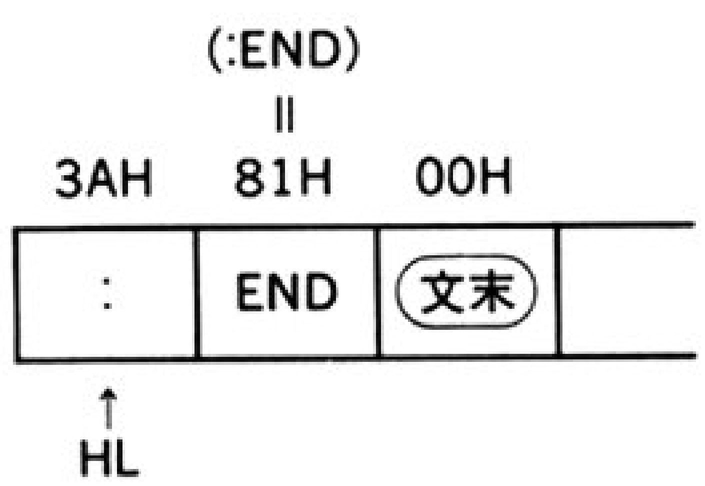


図 2.43 ウォームスタートのための NEWSTT の設定

7.10.3 オートスタート

単純なゲームカートリッジ等では、ROM のヘッダの INIT の場所にプログラムの実行開始アドレスを書くことでそのプログラムを起動できます。しかし、この方法を使うと他のカートリッジの初期設定が行われないため、ディスク等を使えなくなります。

ROMカートリッジのソフトウェアでディスクを使う場合は、カートリッジのINITルーチン実行時に、起動させたいプログラムへのインタースロットコール命令を【H.STKE(0FEDAH)】というフックに書き、リターン命令でシステムへ戻って下さい。すると、全てのカートリッジの初期設定が終わり、ディスクがあればDisk BASICの環境が整ってから、そのフックがコールされ、目的のプログラムを起動できます。

この方法は、ディスクが無い場合でも有効です。



第3部
MSX-DOS

MSXを実用的なコンピュータとして活用するには、ディスクが不可欠です。そこで、ディスクをコントロールするソフトウェア（ディスクオペレーティングシステム）として、MS-DOSに準じた高度な操作性やMS-DOSとのファイル互換性、強力で高速かつ柔軟なファイル処理などを実現したMSX-DOSが開発されました。

第3部では、MSX-DOSについて詳しく説明します。

1章 MSX-DOSの概要

1.1 MSX-DOSの特徴

MSX-DOSは、MSX用のディスクオペレーティングシステムとして、数々の優れた特徴があります。以下で、その特徴について解説します。

1.1.1 MS-DOSに準じた高度な操作性

MSX-DOSは、16ビットマイクロプロセッサ用の汎用OSであるMS-DOSをもとに開発されました。そのため、MSX-DOSを知ることで、MS-DOSの操作にも慣れることができます。

1. MSX-DOSのコマンドの名前と機能はMS-DOSと共通です。そのため、MS-DOSを使用するときも操作の違いで困ることはほとんどありません。
2. ファイルを修正すると、その日付と時刻が自動的にディスクに記録されるので、いつファイルが修正されたかわかります。
3. コマンド行の編集機能を備えています。そのため入力ミスを容易に修正することができ、同じコマンドを続けて入力したり、似たコマンドを入力したりするのが簡単です。
4. 実行するプログラム名を「autoexec.bat」というファイルに書き込んでおくだけで、起動時に自動的に実行する機能を備えています。

1.1.2 MS-DOSとのファイルの互換性

フロッピーディスクのフォーマットがMS-DOSと同じなので、MS-DOSで作成したデータファイルをMSX-DOSで読み書きすることも、その逆も可能です。

1.1.3 強力、高速かつ柔軟なファイル処理

MSX-DOSは以下のような優れたファイル処理を実現しています。

1. MS-DOSと同様に、最大1ギガバイトという大容量のファイルを取り扱うことができます。
2. FATやディレクトリのバッファリングにより、高速なディスクの入出力を実現しています。また、ブロック入出力を利用したプログラムでは、さらに高速な入出力が可能です。
3. ファイルの処理に論理レコードと呼ばれる手法を取り入れて、プログラムの開発を容易にし、ディスクの有効利用を図っています。
4. 周辺機器をディスクファイルと同じように取り扱うことができます。例えば、ファイルからの入力と同じ方法でキーボードから入力したり、ファイルに出力するようにしてプリンタに出力したりすることができます。

1.1.4 Disk BASICとDOS間の関係

MSX Disk BASICはMSX-DOSを利用して動作しています。そのため、BASICの画面編集機能や画面制御命令などの便利な機能を、MSX-DOSから利用することができます。

1. ファイル処理に同じ方法を用いているので、Disk BASICで作成したファイルをMSX-DOSで扱うことも、その逆も可能です。
2. MSX-DOSにはDisk BASICに移行するコマンドが、またDisk BASICにはMSX-DOSに移行するコマンドが用意されているので、相互に往き来することができます。

1.1.5 CP/Mからのプログラムの移植が容易

MSX-DOSのファンクションコールの大部分は、同じ8ビットマイクロプロセッサ用のOSであるCP/Mのファンクションコールと互換性を持っています。したがって、CP/M上の多くのプログラムを移植してそのまま利用することができます。

1.2 この章の表記法

[ ](角型カッコ)　コマンド、ファイル名などが角型カッコに囲まれているときは、それが省略可能な項目であることを示します。

＜ ＞(山型カッコ)　山型カッコは、その中で指示されている項目を入力することを示します。

例

```
copy ＜ファイルスペック1＞ ＜ファイルスペック2＞
rem ＜コメント＞
```

{ }(大カッコ)　大カッコに囲まれているものは、その中にあるもののうちの1つを必ず入力しなければならないことを示します。

|(縦線)　縦線で区切られた項目から1つを選んで入力します。

例

```
VERIFY {ON | OFF}    (ONかOFFのどちらか選ぶ)
```

・・・(省略記号)　直前の項目が必要に応じて何回も繰り返されることを示します。

CAPS(大文字)　大文字は、そのつづりどおりに入力しなければならないコマンドまたはパラメータの一部分であることを示します。

スペース(区切り記号)

「 」はコマンドとパラメータ、またはパラメータとパラメータとの間の区切りを示します。区切り記号にはスペース、タブ、カンマが使用できます。

2章 MSX-DOSの操作

2.1 MSX-DOSの起動と終了

2.1.1 MSX-DOSで使用するディスクの種類

MSX-DOSでは、次の2つのタイプのディスクを使用します。

1. 3.5インチマイクロフロッピーディスク（1DDタイプ）

1DDとは、片面倍密度倍トラック（Single sided、Double density、Double track）の意味で、フォーマット時の記憶容量は360Kバイトです。

2. 3.5インチマイクロフロッピーディスク（2DDタイプ）

2DDとは、両面倍密度倍トラック（Double sided、Double density、Double track）の意味で、フォーマット時の記憶容量は720Kバイトです。

2.1.2 MSX-DOSの起動

MSX-DOSを起動するには、以下の手順で行います。

1. MSXコンピュータとディスプレイ、ディスクドライブ、プリンタ（必要に応じて装備）などの周辺機器が正しく接続されていることを確認します。
2. 周辺機器の電源を入れます。
3. システムディスクをドライブAに挿入します。ここで「ドライブA」とは、次のような

ドライブを指します。

- ■接続しているドライブが1台のときは、そのドライブ。
- ■接続しているドライブが2台のときは、ディスクインターフェイスカートリッジに接続されているドライブ。
- ■なお本体内蔵型のドライブを使用している方は、「2.1.4 4. 複数のドライブを用いるときのドライブ番号」を参照して下さい。

4. MSX コンピュータの電源を入れます。

これで、MSX-DOS がディスクからメモリに読み込まれ、ディスプレイに次のように表示されます。

```
                  MSXDOS.SYS のバージョン番号を表す。
                  ↓
MSX-DOS version 1.03
Copyright 1984 by Microsoft

COMMAND version 1.11
                  ↑
                  COMMAND.COM のバージョン番号を表す。
```

時計機能を装備していない MSX コンピュータの場合、ディスプレイには続いて次のように表示されます。

```
Current date is Sun 84-01-01
Enter new date : ■
                 ↑
                 これがカーソル
```

ここで日付を入力して下さい。入力を間違えたときは、BS(バックスペース)キーを押してカーソルをその場所まで戻し、入力しなおします。

```
Enter new date: 85-9-2

A>
```

時計機能を装備した MSX コンピュータでは、内蔵した時計の時刻と日付が自動的に参照されるので、日付の入力は必要ありません。

```
MSX-DOS version 1.03
Copyright 1984 by Microsoft

COMMAND version 1.11
A>
```

MSX-DOSが起動すると、ディスプレイに「A>」という記号が表示されます。この記号はMSX-DOSの入力促進記号(プロンプト)で、これが表示されている時はコンピュータが入力を受付ける状態であることを示します。この状態を「MSX-DOSのコマンド入力待ち状態にある」といい、プロンプトに続くコマンド行に様々なコマンド名を入力し、プログラムを実行することができます。

プロンプトにある「A」は、現在MSX-DOSが優先して使用するドライブ(デフォルトドライブ)の名前で、「B>」ならばデフォルトドライブがドライブBであることを示します。ここで「d:」(dはドライブ名)を入力すると、デフォルトドライブを変更することができます。この結果プロンプトのドライブ名は指定したものに変わります。たとえばデフォルトドライブをAからBに変更するとき、次のように入力します。

```
A>B:
B>
```

なお、MSX-DOSを起動した時点でのプロンプトは「A>」です。

2.1.3 MSX-DOSの終了

MSX-DOSでの処理が終わったら、次のような手順でMSX-DOSを終了します。

1. 最後に実行したコマンドまたはプログラムの処理が終了していることを確認します。処理が終っていれば、ディスプレイにはプロンプトが表示されています。
2. ディスクドライブからディスクを取り出します。
3. MSXコンピュータの電源を切ります。
4. 各周辺機器の電源を切ります。

最後に実行したコマンドまたはプログラムの処理が終了していないうちにコンピュータの電源を切ると、ディスクに記録されていたファイルが正しく読み出せなくなってしまう可能性があるので注意して下さい。

2.1.4 ハードウェア構成による使用上の違い

MSX-DOSは、時計機能の有無、MSX1かそれともMSX2 (MSX2+) か、ディスクドライブが何台接続されているかなど、コンピュータを構成するハードウェアの違いによってその使用法が異なります。

1. 時計機能

時計機能はMSX1ではオプションで、MSX2とMSX2+では標準装備です。

時計機能を装備したMSXは、日付、時刻を自動的に更新し、閏年、月の大小、曜日なども管理します。また、MSX-DOSの起動時には、時計が持つ日付と時刻が自動的に参照されるので、日付の入力は必要ありません。ただし、長時間電源を切ったままにしておくと時計の動作が停まり、正しい時刻と日付を示さなくなることがあります。そのようなときは、「date」コマンドと「time」コマンドを使って、正しい日付と時刻に修正して下さい。なお「date」コマンド、「time」コマンドについてはコマンドの解説を参照して下さい。

2. MSXとMSX2 (MSX2+)

MSXとMSX2では、ディスプレイの最大表示文字幅が異なり、MSXでは40字、MSX2とMSX2+では80字になっています。なお、MSX2とMSX2+の機能として、文字の表示幅、画面の前景色、背景色、周辺色、ビープ音などを、起動時に自動的に設定することができます。これらは、BASICで好みの値に設定し「SET SCREEN」命令、および「SET BEEP」命令でバッテリバックアップしたSRAM内に記憶され、起動時にはこの値が用いられます。

3. 仮想ドライブ機能

MSX-DOSは、1台のドライブでも使用できます。これはMSX-DOSが、1台のドライブをあたかも2台のドライブが接続されているかのごとく動作させる、仮想ドライブ機能(2ドライブシミュレータ) を備えているためです。

表3.1に、内蔵型のディスクを持たず、ディスクインターフェイスカートリッジが1台のときの、仮想ドライブ機能の有無を示します。

表3.1 仮想ドライブ機能の有無

ディスクインターフェイスカートリッジ	ドライブ数	仮想ドライブ機能
1台	1台	有
1台	2台	無

■仮想ドライブ機能が動作しているときの操作方法

MSX-DOS は、ディスクインターフェイスカートリッジ1つに1台のドライブしかつながっていないときに、標準で仮想ドライブ機能が動作します。

MSX-DOS は仮想ドライブ（ここではドライブ B）を扱うコマンドを受け取ると、そのコマンドの処理中に次のようなメッセージを表示します。

```
Insert diskette for drive B:
and strike a key when ready
```

（ドライブ B 用のディスクを入れ、どれかキーを押して下さい）

ここで、その処理に必要なもう1枚のフロッピーディスクと入れ換えます。すると、MSX-DOS はそのフロッピーディスクをもう1台の仮想ドライブとみなして処理を行います。その処理が終って、もとのディスクに対する処理が必要になると、再びディスプレイに次のようなメッセージを表示します。

```
Insert diskette for drive A:
and strike a key when ready
```

（ドライブ A 用のディスクを入れ、どれかキーを押して下さい）

このメッセージにしたがって処理を行えば、2台のドライブを必要とする処理を、1台のドライブで行うことができます。このように仮想ドライブ機能は、MSX-DOS を1台のドライブで運用する上で、重要な役割を果しています。

■仮想ドライブ機能を動作させないとき

2台のディスクインターフェイスカートリッジを用いて、それぞれに1台のドライブを接続したときなど、仮想ドライブ機能を用いるとかえって操作が複雑になることがあります。そのときには、起動時にビープ音が鳴るまで CTRL キーを押し続けて、仮想ドライブ機能を動作させないようにして下さい。

4. 複数のドライブを用いるときのドライブ番号

図 3.1～3.6 に、複数のドライブを接続するとき、どのようにドライブ番号が割り当てられるかを示します。カッコ内は仮想ドライブ機能が動作している場合です。起動後のドライブ番号がこの解説と異なるときは、使用しているハードウェアのマニュアルを参照して下さい。

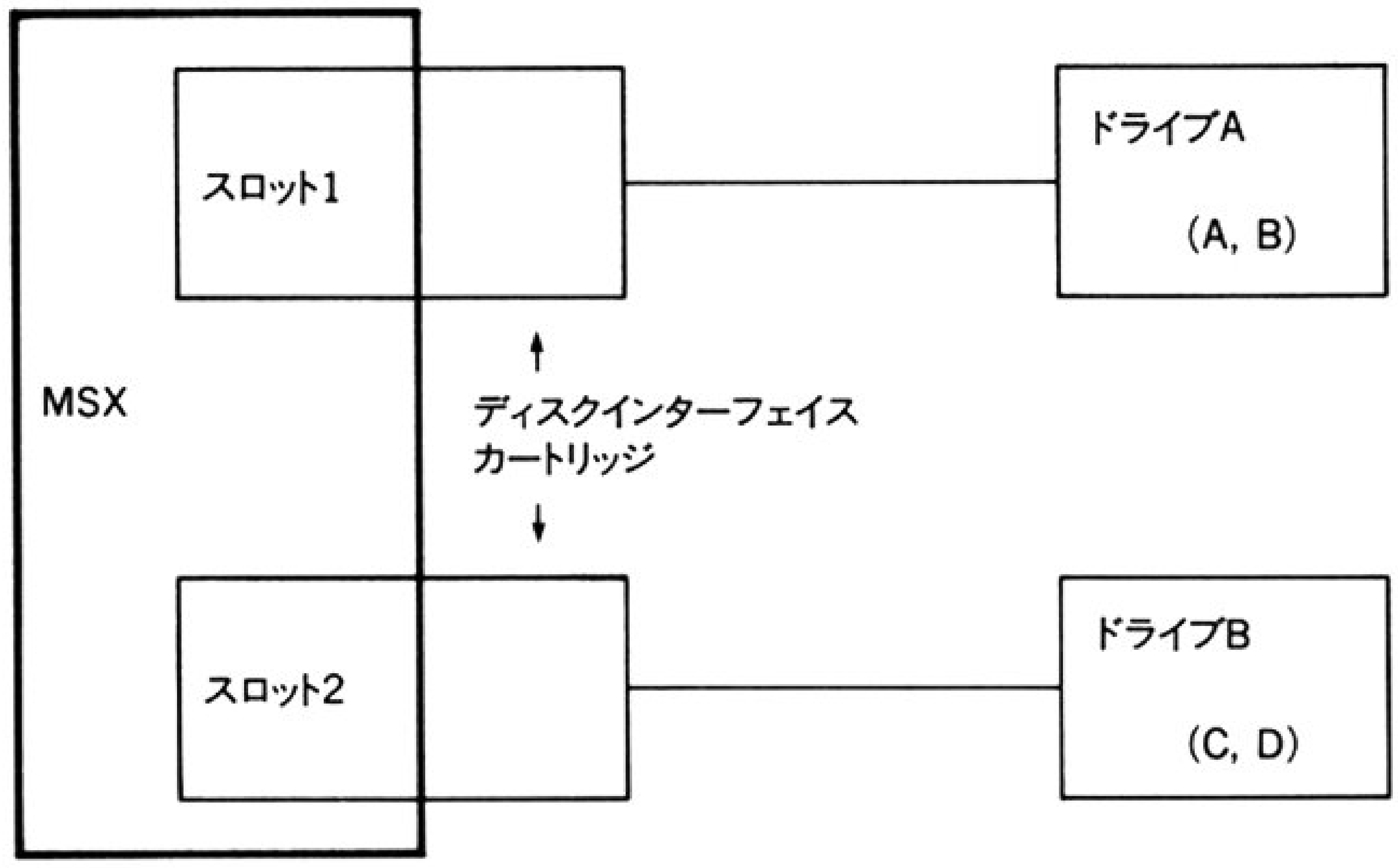


図3.1　2台のディスクインターフェイスにドライブを1台ずつ接続

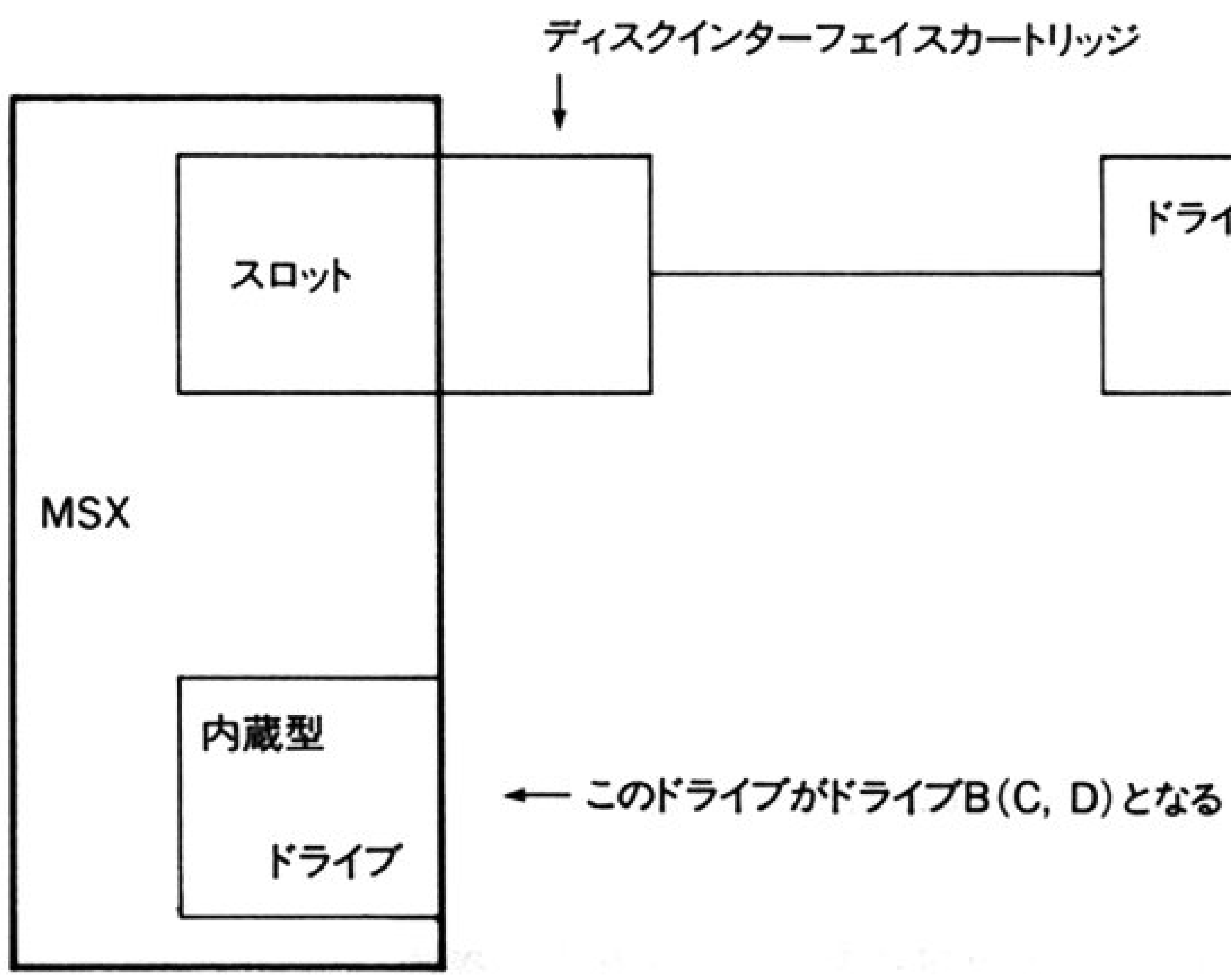


図3.2　内蔵型ドライブ1台＋ディスクインターフェイスにドライブを1台接続

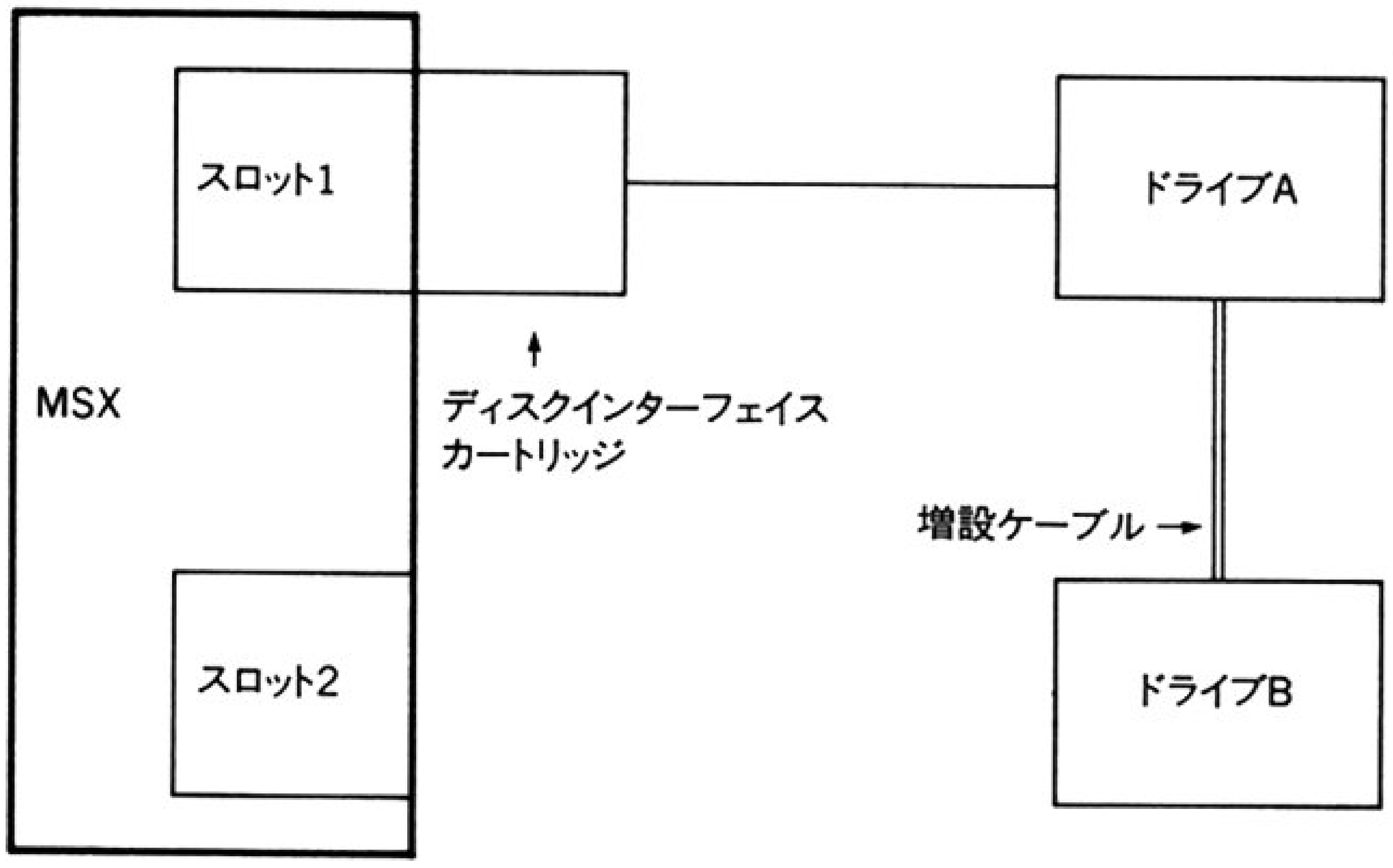


図 3.3　1 台のディスクインターフェイスに外付け型ドライブを 2 台接続

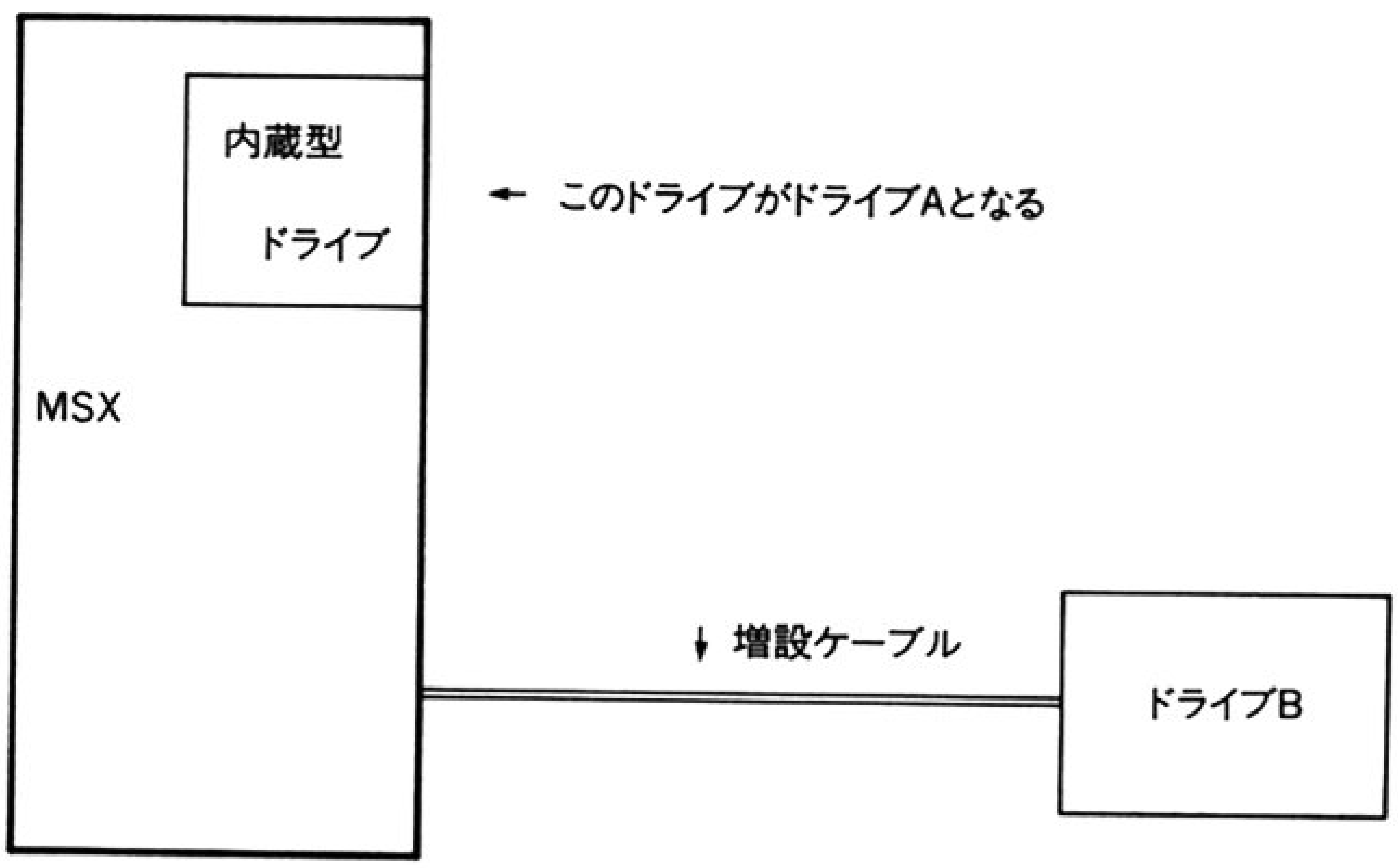


図 3.4　内蔵型ドライブ 1 台＋増設ケーブルにより外付け型ドライブを 1 台接続

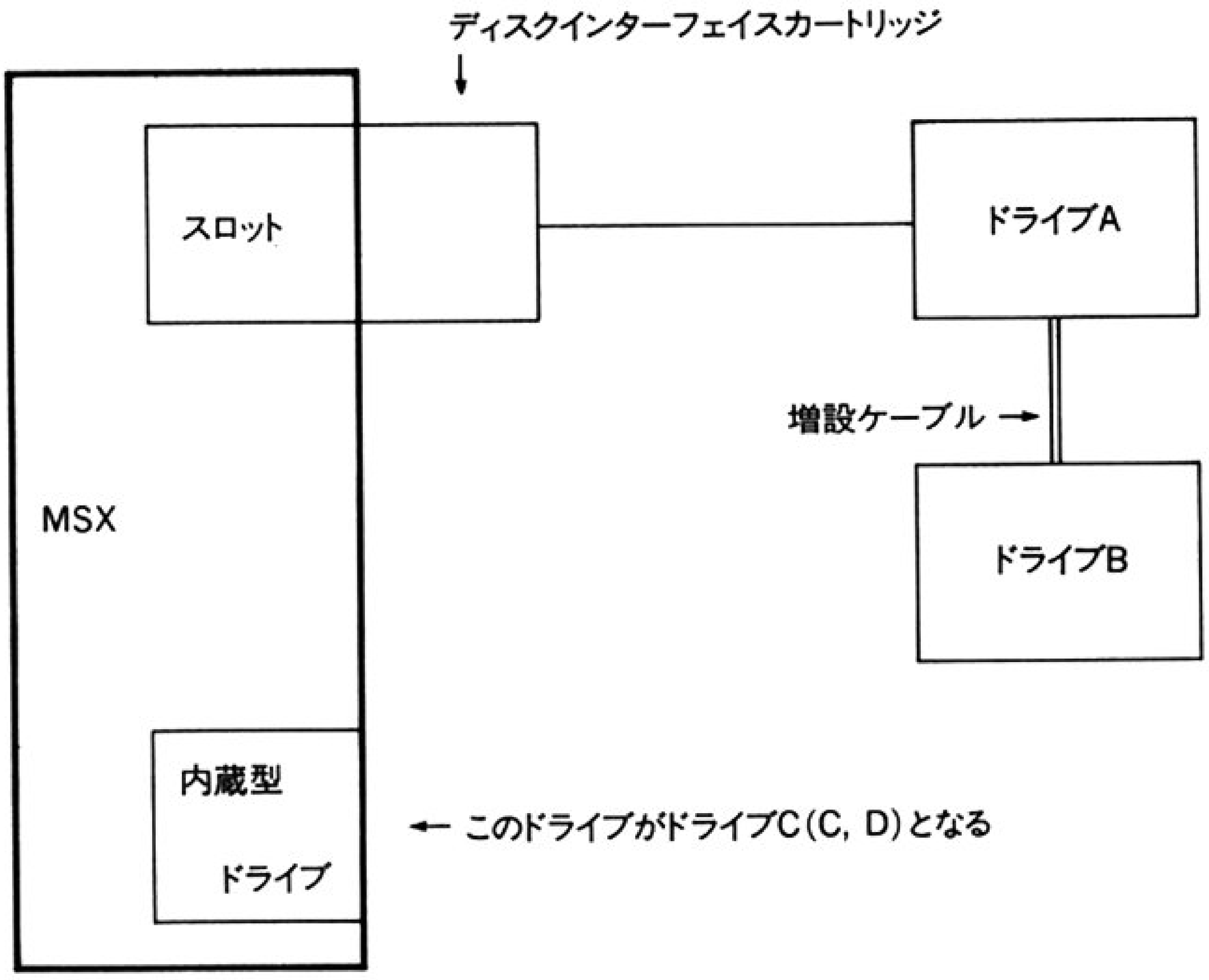


図3.5　内蔵型ドライブ1台＋ディスクインターフェイスに外付け型ドライブを2台接続

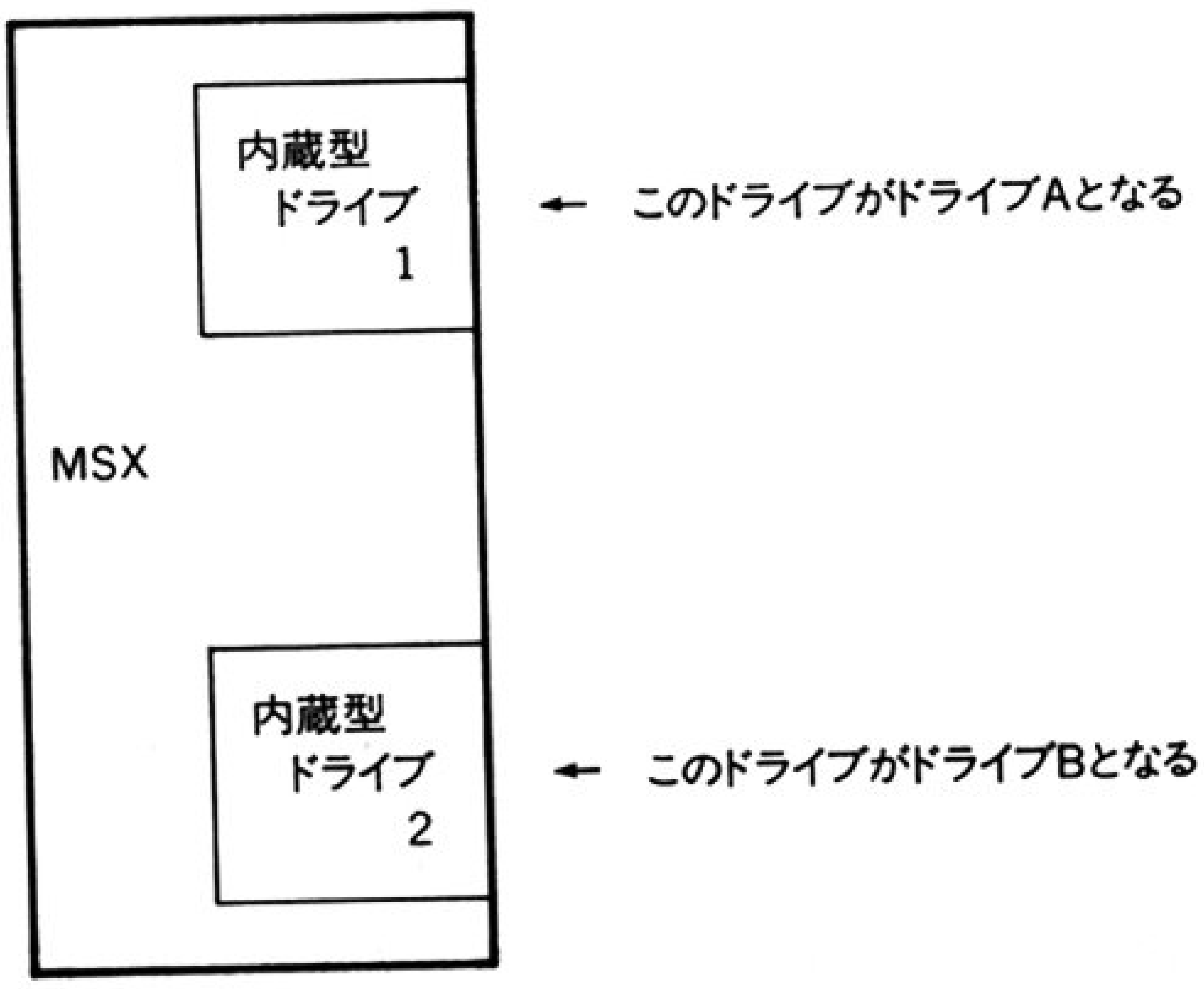


図3.6　2台の内蔵型ドライブ

2.2 ファイル

2.2.1 ファイルの概要

コンピュータは、外部記憶装置としてフロッピーディスクドライブ、クィックディスク、データレコーダ（テープレコーダ）や紙テープ装置などを使用します。これらの外部記憶装置に記録されたプログラムやデータを総称してファイルといいます。

DOSは、このようなファイルの登録、既存のファイルへの追加、修正、ファイルの削除などの処理を行う機能を持っています。このような機能（それを実現するプログラム）を、ファイルマネージメントシステム、または単にファイルシステムと呼んでいます。

DOSが扱うファイルとは、普通ディスクに記録されるディスクファイルです。しかし、MSX-DOSでは、プリンタなどの周辺機器をもファイルとして管理しています。このため、周辺機器を扱うためのプログラムをわざわざ開発する必要がなく、周辺機器の取り扱いが簡単になります。各周辺機器には特殊なファイル名が割り当てられており、このファイル名はディスクファイル名と同様にコマンド行で使用します。

2.2.2 ディスクファイル

MSX-DOSではディスクファイルの管理を、それぞれのディスクに用意されているFAT(File Allocation Table）とディレクトリ（Directory）を用いて行います。FATとは、ディスクでのファイルの配置状況を記録した地図のようなものです。また、ディレクトリは、ファイル名とそのファイルに関する情報を記録した索引です。DOSを利用するときはこのうちディレクトリにあるファイル名だけを意識していればよく、実際のファイル管理はすべてDOSが行います。

以下のように「dir」コマンドを用いてデイレクトリを見ると、そのディスクにどんなファイルがあり、そのファイルの大きさといつ修正されたか、そしてディスクの残り容量がわかります。

```
A>dir
MSXDOS    SYS     2432 85-08-23  9:29p
COMMAND   COM     6656 85-09-02 10:10p
         2 files  352256 bytes free
A>
```

1. ファイル名

ファイル管理の中心となるものは、そのファイルに付けられたファイル名です。ファイル名は、1〜8文字のファイル名と1〜3文字の拡張子からなっていますが、拡張子は必ずしも必要ではありません。また、ファイル名と拡張子の間はピリオド「.」で区切ります。ファイル名として使用できる文字は次のとおりです。

A〜Z　0〜9　$　&　#　@　!　%　'　(　)
ー　{　}　〜　ひらがな　カタカナ

なお、アプリケーションプログラムによっては、ファイル名として一部の文字が使えないことがあります。そのときにはA〜Z、0〜9を使用して下さい。

上記の文字は、ファイル名や拡張子のどの部分にも使うことができます。その際、アルファベットの小文字は、MSX-DOSによって大文字に変換されます。

拡張子は、ファイルの種類を区別しやすくするために使います。拡張子には、MSX-DOSでその使用が決っているものや、アプリケーションによっては習慣的に特定の拡張子を用いるものがあります。そのようなときには、必ずその拡張子を使って下さい。

■ MSX-DOSでその使用が決っている拡張子

拡張子	ファイルの種類	例
.BAT	バッチファイル	AUTOEXEC.BAT
.COM	コマンドファイル	COMMAND.COM
.SYS	システムファイル	MSXDOS.SYS

■習慣的に使用する拡張子

拡張子	ファイルの種類
.ASM	アセンブラソースファイル
.BAK	バックアップファイル
.BAS	ベーシックプログラムファイル
.C	C ソースファイル
.COB	COBOL ソースファイル
.CRF	クロスリファレンスファイル
.DAT	データファイル
.FIF	FORTH ソースファイル
.FOR	FORTRAN ソースファイル
.HEX	インテル HEX オブジェクトファイル
.LIB	アセンブラライブラリファイル
.M80	アセンブラライブラリファイル
.MAC	アセンブラソースファイル
.PAS	PASCAL ソースファイル
.PLI	PL/I ソースファイル
.PRN	リスティングファイル
.REL	リロケータブルオブジェクトファイル

なお、アプリケーションによっては、使う拡張子が決っているものがあります。

2. ワイルドカード文字

ワイルドカード文字とは、ファイル名を指定する際に、任意の1字または文字列に対応して、その文字または文字列の代りに用いることができる省略記号のことです。ワイルドカード文字を用いると、ファイルを指定する際にいくつかのファイルをまとめて指定することができます。ワイルドカード文字には、任意の1字に対応する「?」(クエスチョンマーク)と、任意の文字列に対応する「*」(アスタリスク)の2種類があります。

「dir」コマンドを例に、ワイルドカード文字の使い方を解説します。「dir」コマンドにワイルドカード文字を用いれば、限定されたファイルについての情報を表示させることができます。

ここでは、ドライブAに次のようなディスクが入っていたとして述べます。

```
A>dir
ABCDE     EXT     1664 85-07-17  9:18a
ABXDE     EXT      256 85-07-26  6:56p
ABYDE     EXT     1280 85-07-31  3:48p
ABCZDE    COM     9984 85-08-09 12:08p
XYZ       COM     8320 85-08-25  3:39a
        5  files   337920 bytes free
A>
```

■「?」(クエスチョンマーク)

「dir」のパラメータに「ab?de.ext」というファイル名を指定したとします。この場合、「dir」は以下に示すようにファイル名が全体で5文字でABで始まりDEで終り、拡張子が「.EXT」のすべてのファイルを表示します。

```
A>dir ab?de.ext
ABCDE      EXT      1664 85-07-17   9:18a
ABXDE      EXT       256 85-07-26   6:56p
ABYDE      EXT      1280 85-07-31   3:48p
         3  files    337920 bytes free
A>
```

なお、「?」は、拡張子の中でも同様に使うことができます。

■「＊」(アスタリスク)

ファイル名または拡張子の中に「＊」があると、「＊」はその位置以降の文字列を「?」に置き換えられます。つまり、「＊」は連続した「?」の略記と同じです。次の2つのコマンドは全く同じ意味で、そのディスクに記録されているすべてのファイルを表示します。

```
A>dir *.*
A>dir ????????.???
```

次のコマンドは、拡張子が「.COM」の全てのファイルを表示します。

```
A>dir *.com                              ←「dir ????????.com」と同じ
ABCZDE     COM      9984 85-08-09  12:08p
XYZ        COM      8320 85-08-25   3:39a
         2  files    337920 bytes free
A>
```

次のコマンドは、ファイル名がABCで始まる全てのファイルを表示します。

```
A>dir abc *.*                            ←「dir abc?????.???」と同じ
ABCDE      EXT      1664 85-07-17   9:18a
ABCZDE     COM      8320 85-08-25  12:08p
         2  files    337920 bytes free
A>
```

3. ドライブ指定

MSX-DOSはファイルをディスクドライブごとに管理しています。そのため、違うドライブに同じ名前のファイルがあっても、問題はありません。

MSX-DOSが管理するドライブの数は最大8台で、それぞれA、B、C～Hのアルファベットのドライブ名を持っています。ドライブを指定するときには、この文字の後ろに「：」（コロン）を付け、「d:」というように指定します。この文字とコロンの組み合わせを、「ドライブ指定」といいます。

また、ドライブ指定、ファイル名、拡張子をこの順で続けたものを、特に「ファイルスペック」といいます。ファイルスペックの書式は以下のとおりです。

［<ドライブ指定>］<ファイル名>［.<拡張子>］

例

A:BCDEFGHI.BAS

なお、デフォルトドライブにあるファイルを指定するときは、ドライブ指定は省略することができます。

2.2.3 特殊なファイル名

MSX-DOSでは周辺機器に特殊なファイル名を割り当て、各周辺機器をディスクファイルと同じようにコマンド行で取り扱えるようにしています。この特殊なファイル名を「デバイスファイル名」といいます。デバイスファイル名には次のものがあります。

デバイスファイル名	機　能
AUX	データレコーダなどの補助入出力装置との入出力や他のコンピュータとの通信のために用意してあります。起動時は、NULと同じ機能に設定してあります。
CON	キーボードからの入力や、ディスプレイへ出力するときに使用します。
NUL	コマンドがその構文上、入出力のファイル名を要求していても、特にファイルを作らないときに使用します。
PRN	プリンタに出力するときに使用します。LSTもこれと同じです。

デバイスファイル名を用いた例を以下に2つあげます。

```
A>copy con autoexec.bat
```

キーボードからの入力を「autoexec.bat」というファイル名でデフォルトドライブ上に作ります。

A>copy test.mac prn

「test.mac」というファイルの内容をプリンタに打ち出します。

なお、デバイスファイルを使用するとき、たとえドライブ指定や拡張子が指定されていても、デバイスファイル名が優先します。したがって、CON.LST はファイルにはなりません。例えば、下のようなコマンドを入力しても、CON.LST というディスクファイルは作られず、SAMPLE.MAC の内容がディスプレイに表示されます。

A>copy sample.mac con. 1st

2.3 コマンドの概要

2.3.1 コマンドとは

1. コマンドの概念

BASIC などのプログラミング言語では、コマンドは1つのプログラムを構成する部分品でした。しかし MSX-DOS でいうコマンドとは、プログラミング言語のコマンドとは違い、プログラムそのものです。コマンドは、プロンプトに続くコマンド行にその名前を入力することで実行されます。

MSX-DOS で実際にコマンドの実行を受け持つのはシステムコマンドファイルである「COMMAND.COM」です。すべてのコマンド（プログラム）は、「COMMAND.COM」が解析し、実行されます。

2. コマンドの書式

コマンドは、コマンド名とそれに続くパラメータからなっています。コマンドによって、パラメータが必要なものとそうでないものがあります。また、コマンド名とパラメータ、パラメータ相互の間は、スペース、タブまたはカンマによって区切ります。コマンドの書式は以下のようになります。

A>[＜ドライブ指定＞]＜コマンド名＞[　＜パラメータ＞]・・・

これは、＜ドライブ指定＞で指定されたドライブにある＜コマンド名＞というコマンドを実行せよ、ということです。

2.3.2 コマンドの種類

コマンドには、MSX-DOSの内部にあらかじめ用意されている内部コマンドと、ディスクにファイルとして記録されていて、コマンド名が入力されるたびにメモリに読み込まれて実行される外部コマンド、MSX-DOSのバッチ処理機能により自動的に複数の内部コマンド、外部コマンドを実行するバッチコマンドの3種類があります。

1. 内部コマンド

MSX-DOSには、以下の13種類の内部コマンドがあります。それぞれのコマンドの詳細については、「2.4 コマンド一覧」を参照して下さい。

BASIC	DIR	REN	VERIFY
COPY	FORMAT	REM	
DATE	MODE	TIME	
DEL	PAUSE	TYPE	

2. 外部コマンド

アセンブラやコンパイラなどを利用して作成した「.COM」という拡張子を持つファイルが外部コマンドです。システムディスクの「COMMAND.COM」もこれにあたります。

外部コマンドはファイル名をコマンド行から入力すると、ディスクから読み込まれて実行されます。なお、コマンドファイルがデフォルトドライブにないときは、コマンド名にドライブ指定をつけるか、デフォルトドライブをコマンドファイルがあるドライブに変更しなければなりません。

以下に外部コマンドとデフォルトドライブの関係を示します。例えば、ドライブA、Bに、それぞれ次のようなディスクが入っているとします。

```
A>dir
TEST       COM      384 85-07-25   6:16p
NEWDISK    BAT       78 85-08-03  10:02p
        2 files  360448 bytes free
A>dir b:
CAT        COM     8320 85-08-20   3:04p
        1 files  353280 bytes free
A>test                                    ←「test」を入力
```

TESTというコマンド（プログラム）を実行

```
A>cat                                  ←「cat」を入力
Bad command or file name               ←CAT.COMがデフォルトドライブであ
                                         るドライブA上になかったので、エラー
                                         メッセージが表示された。

A>b:cat
```

CATを実行

```
A>b:                                   ←デフォルトドライブをドライブBに変更
B>cat                                  ←「cat」を入力
```

CATを実行

```
B>
```

3. バッチコマンド

バッチコマンドとは、拡張子が「.BAT」であるバッチファイルに記録した一連のコマンドで、バッチファイル名をコマンド行から入力すれば、記録されているコマンドが次々と実行されます。バッチコマンドを利用することで、個々のコマンドをいちいち入力することなく、一連の処理を自動的に行うことができます。このため、アセンブルやコンパイルなど、同じコマンドを使って何回も処理を行うときには、非常に便利な機能です。

バッチファイルがデフォルトドライブにないとき、バッチファイル名の前にドライブ指定をつけるか、デフォルトドライブを変更しなければならなりません。これは、外部コマンドのときと同じです。

バッチファイル「NEWDISK.BAT」が次のような内容であるとします。ファイルの内容をディスプレイに表示させるには、「type」コマンドを使います。

```
A>type newdisk.bat
rem This is file NEWDISK.BAT
pause Insert new disk in drive b:
format
dir b:

A>
```

このバッチファイル名を入力すれば、バッチファイルの内容が次々と実行されます。

```
A>newdisk
A>rem This is file NEWDISK.BAT
A>pause Insert disk in drive B:
Strike a key when ready             ←ドライブBに新しいディスクを入れたらどれか
                                      キーを押す

A>format
Drive name?(A, B) b                 ←ドライブBのディスクをフォーマット
Strike a key when ready             ←どれかキーを押す
Format complete                     ←フォーマットが終了すると表示される
```

```
A>dir b:
File not found                    ←ドライブBのディスクにはファイルがない
A>
```

2.4 コマンド一覧

ここではMSX-DOSの内部コマンドについて解説します。コマンドはアルファベット順になっています。

コマンド名	機　能	ページ
BASIC	Disk BASIC に移行します。	355
COPY	ファイルをコピーまたは連結します。	356
DATE	日付を表示、変更します。	360
DEL (ERASE)	ファイルを削除します。	362
DIR	ディレクトリを表示します。	363
FORMAT	ディスクを初期化します。	364
MODE	文字表示幅を設定します。	366
PAUSE	バッチコマンドの実行を一時停止します。	367
REM	バッチファイルにコメントを書き込みます。	368
REN (RENAME)	ファイル名を変更します。	368(369)
TIME	時刻を表示、変更します。	369
TYPE	ファイルの内容を表示します。	370
VERIFY	書き込みの際の検査の有無を設定します。	371

BASIC

機 能

Disk BASICを起動します。

書 式

BASIC [<ファイルスペック>]

解 説

MSX-DOSが起動されている状態からDisk BASICを起動したいときは、「basic」コマンドを用います。<ファイルスペック>によってDisk BASICのプログラムファイルを指定すると、Disk BASICを起動した後、そのプログラムを読み込んで実行します。

Disk BASICからMSX-DOSへ移行するには「_system（またはcall system)」命令を実行します。

例

Disk BASICを起動し、アスキーセーブされたBASICプログラムの「FUNCKEYS.INI」を実行します。このプログラムは、Disk BASICでファンクションキーを設定して、再びMSX-DOSに戻るものです。

```
A>type funckeys.ini
100 KEY 1, "dir"
110 KEY 2, "ren"
120 KEY 3, "copy"
130 KEY 4, "del"
140 KEY 5, "type"
150 KEY 6, "time"+CHR$(13)
160 KEY 7, "date"+CHR$(13)
170 KEY 8, "format"+CHR$(13)
180 KEY 9, "verify"
190 KEY 10, "basic"
200 CLS:CALL SYSTEM

A>basic funckeys.ini
```

注 意

MSX-DOSとDisk BASICではメモリ構成が異なり、「basic」コマンドの実行によってメモリが切り換えられます。このため、メモリを介してMSX-DOSとDisk BASICとの間でデータをやりとりすることはできません。

COPY

機　能

ファイルを複写（コピー）、または連結（付加）します。

書 式 1

COPY [/A | /B] <ファイルスペック 1>[/A | /B] [<ファイルスペック 2>[/A | /B]]

書 式 2

COPY[/A | /B] <ファイルスペック 1>[/A | /B]+<ファイルスペック 2>[/A | /B][•••+<ファイルスペック n>[/A | /B]] [<ファイルスペック>[/A | /B]]

解　説

<ファイルスペック>の指定にはワイルドカード文字を使うことができます。普通、「copy」コマンドは、ファイルを複写するために使用します。

■同じディスクへのファイルの複写

以下は、「MSXDOS.SYS」というファイルを「MSXDOS.BAK」というファイル名で同じディスクに複写するときのコマンド行の例です。

```
A>copy msxdos.sys msxdos.bak
```

ここで<ファイルスペック 1>と<ファイルスペック 2>は、必ず違うファイル名でなければなりません。同じファイル名を指定すると、「copy」コマンドは中断され（ファイルをそれ自身にコピーすることは許されません）、次のようなエラーメッセージが表示されます。

```
A>copy msxdos.sys msxdos.sys
File cannot be copied onto itself
0 files copied
A>
```

■違うディスクへのファイルの複写

このとき、<ファイルスペック 2>には4つの形があります。

1. 省略されたとき

コピーはデフォルトドライブに対して行われ、コピー後のファイルには<ファイルス

ペック 1>と同じ名前がつけられます。次の例では、ドライブBの「msxdos.sys」がデフォルトドライブであるドライブAにコピーされます。

```
A>copy b:msxdos.sys
```

<ファイルスペック 1>がデフォルトドライブ上のもので、<ファイルスペック 2>が省略されていると、「copy」コマンドは中断され（ファイルをそれ自身にコピーすることは許されません）、次のようなエラーメッセージが表示されます。

```
A>copy msxdos.sys
File cannot be copied onto itself
0 files copied
A>
```

2. ドライブ指定のみのとき

指定されたドライブに同じ名前でコピーします。

```
A>copy a:msxdos.sys b:
```

3. ファイル名だけのとき

指定されたファイル名でデフォルトドライブにコピーします。

```
A>copy b:msxdos.sys msxdos.bak
```

4. 完全なファイルスペックを指定したとき

指定されたドライブへ指定された名前でファイルをコピーします。

```
A>copy msxdos.sys b:msxdos.sys
```

■「copy」コマンドのスイッチ

/Aスイッチと/Bスイッチでコピーモードを指定します。

/A　アスキーモード（Ascii mode）の意味で、処理しているファイルの中にエンドオブファイル文字(EOF、ファイルの最後に付けられる)があると、そこでファイルが終っているとみなして処理を終了します。

/B　バイナリモード（Binary mode）の意味で、EOFがあっても処理を続けます。

<ファイルスペック>にスイッチ</A>または</B>がついているとき、そのスイッチは直前の<ファイルスペック>から次のスイッチが指定されるまで有効です。/Aが指定されている間に読み取られるファイルは、EOF以降がコピーされません。
コピー先のファイル名につけられるスイッチは、EOFがファイルの終りに置かれるかどうかを決定します。/Aスイッチが指定されていると、ファイルの書き込みではEOFが

1つだけつけられます。したがって、アスキーファイルを

```
A>copy ab.mac/B cd.mac/A
```

というコマンドでコピーすると、

■もとのファイルのEOFは/Bスイッチのため除去されず、

■コピー先のファイルの/Aスイッチにより余分なEOFが付加されるので、

結果として2つのEOFが付いたファイルができることになります。
次はファイルを連結するときにスイッチがついた例で、実行ファイル「PROG.COM」に、アスキーファイルである定数データ「ERRS.TXT」を連結するときののもです。結果として、必要なのがバイナリファイルとすると、終りのEOFは必要ないのでスイッチをこのように用います。

```
A>copy prog.com/b+errs.txt/a newprog.com/b
```

スイッチを指定しないで複写すると、バイナリモードでコピーされます。

```
A>copy/a ab.txt cd.txt
```

このとき、「AB.TXT」の途中にEOFが含まれていると、その時点でコピーは終了するので、「CD.TXT」の大きさは「AB.TXT」より小さくなることがあります。なお、このとき「CD.TXT」には、EOFが最後の文字としてついています。

■ファイルの連結

「copy」コマンドではファイルを連結することもできます。連結は、「copy」コマンドのパラメータとして＜ファイルスペック＞を「＋」でつなげて指定します。
以下の例では、「AB.TXT」、「CD.TXT」、「EF.TXT」を連結し、その結果を「GH.CRP」という名前でドライブAに書き込みます。
連結は、通常アスキーファイルで行います。バイナリファイルを連結するときは、ファイルの中にEOFが含まれていると、その時点でそのファイルのコピーを打ち切り、次のファイルを続けて連結コピーします。そのため、バイナリファイルを連結するときは/Bスイッチによりバイナリモードに切り換えて行います。

```
A>copy ab.txt+cd.txt+ef.txt gh.txt
```

以下の例では、「AB.COM」に「CD.COM」を連結し、「EF.COM」というファイルを作ります。

```
A>copy/b ab.com+cd.com ef.com
```

次の例は、ドライブBにある「TEST1.TXT」に、デフォルトドライブにある「TEST2.TXT」を付加するときのコマンド行です。

A>copy b:test1.txt+test2.txt b:test1.txt

ただし、もとになるファイルと付加するファイルが双方ともデフォルトドライブ上にあるときは、最後のパラメータを省略することができます。

■ワイルドカード文字の使用例

1. 複写（コピー）

ワイルドカード文字を利用すれば、ディスクのバックアップや特定のファイル名または拡張子を持つファイルのみをコピーすることができます。
次の例では、デフォルトドライブであるドライブAのすべてのファイルをドライブBにコピーします。

A>copy *.* b:

次の例では、デフォルトドライブにある拡張子が「.COM」であるファイルのすべてを、ドライブBにコピーします。

A>copy *.com b:

次の例では、ドライブBにあるファイル名が3字以下であるファイルのすべてを、デフォルトドライブ（ドライブA）にコピーします。

A>copy b:???.*

2. 連結（付加）

ワイルドカード文字を利用すれば、特定のファイル名または拡張子を持つファイルをすべて連結し、1つのファイルを作るなどの処理ができます。
次の例では、拡張子が「.LST」であるすべてのファイルを連結して、「CONBIN.PRN」という名前のファイルを作ります。

A>copy *.lst conbin.prn

また、以下のようにいくつかのファイルを個々に連結したり、または1つのファイルに連結したりすることもできます。例えば、ドライブAに次のようなファイルがあるとします。

```
A>dir
FILE1       LST     ...............
ABCDE       LST     ...............
FILE1       REF     ...............
ABCDE       REF     ...............
             .
             .
```

このとき、次のようなコマンドを入力すると、「FILE1.LST」と「FILE1.REF」が連結されて「FILE1.PRN」という名前のファイルが作成され、「ABCDE.LST」と「ABCDE.REF」が連結されて「ABCDE.PRN」という名前のファイルができます。

```
A>copy *.lst+*.ref *.prn
FILE1       LST
ABCDE       LST
          2  files copied
A>dir
FILE1       LST     ················
ABCDE       LST     ················
FILE1       REF     ················
ABCDE       REF     ················
FILE1       PRN     ················
ABCDE       PRN     ················
            .
            .
```

次の例では、「*.LST」に該当するすべてのファイルを連結してから、「*.REF」に該当する全ファイルを連結し、「COMBIN.PRN」というファイルを作ります。

```
A>copy *.lst+*.ref conbin.prn
```

注 意

連結した結果作られるファイル名として、すでにディスクにあるファイルの名前を指定すると、そのファイルに連結するファイルの内容が重ね書きされます。結果としてもとのファイルの内容は書き変えられてしまうので、MSX-DOS は警告メッセージを表示します。

```
A>copy *.lst all.lst
Content of destination lost before copy
          1  file copied
A>
```

次の例では、拡張子が「.LST」である全ファイルを、「ALL.PRN」というファイルに付加します。

```
A>copy all.prn+*.lst
```

DATE

機 能

日付の表示、変更を行います。

書 式

DATE [<年>-<月>-<日>]

解 説

パラメータなしで入力したときは次のように表示されます。

```
A>date
Current date is www yy-mm-dd
Enter new date:
```

www、yy、mm、dd は内蔵の時計が示す日付をもとに表示されます。

www　曜日を示します。曜日は日付から自動的に計算されます。表示されるのは次のうちの1つです。

Sun、Mon、Tue、Wed、Thu、Fri、Sat

yy　西暦年を示します。下2桁が表示されます。

mm　月を示します。2桁の数字です。

dd　日を示します。2桁の数字です。

日付を変更する必要がないときは、ここでリターンキーを押します。
パラメータとして日付を入力すると、そのまま新しい日付がセットされます。このときは、メッセージは表示されません。

```
A>date 85-9-2
```

日付の入力には数字だけが使用できます。文字は使えません。指定できるパラメータの範囲は次のとおりです。

年　1980～2079
　　または 80～99（1980～1999 と解釈される）
　　または 00～79（2000～2079 と解釈される）

月　1～12（01～12）

日　1～31（01～31）

年月日の入力は、ハイフン（-）かスラッシュ（/）で区切って下さい。
日付は時刻とともに MSX-DOS が管理するので、時刻が24時になると自動的に更新されます。また、月の大小や閏年もチェックされ、正しく変更されます。ただし、時計機能がない MSX のときは更新はされません。
パラメータや区切り記号が誤っているときは、次のメッセージが表示されるので、もう一度正しく入力して下さい。

```
Invalid date        ←日付の指定が違います
Enter new date      ←日付を入力して下さい
```

注　意

本体のROM BIOSが日本版以外のものでは、日付の表示と入力の形式が次のように変わります。

インターナショナル版	mm-dd-yyyy
ヨーロッパ版	dd-mm-yyyy

DEL

機　能

<ファイルスペック>で指定されたファイルを削除します。

書　式

DEL <ファイルスペック>

解　説

<ファイルスペック>で指定されたファイルを削除します。
次の例は、デフォルトドライブからファイル「TEST.MAC」を削除します。

```
A>del test.mac
```

「del」コマンドでは、削除するファイルの指定にワイルドカード文字が使用できます。例えば、拡張子が「.TXT」のファイルをすべて削除するときは、次のように入力します。

```
A>del *.txt
```

ワイルドカード文字の詳細については、「2.2.1 ファイルの概要」を参照して下さい。
ファイルの指定に「*.*」を用いると、ディスクにあるすべてのファイルを削除することになります。MSX-DOSは実行してもよいかどうかを確認するため、次のようなメッセージを表示します。全ファイルを削除してもよいときは、ここで「y」を入力します。

```
A>del *.*
Are you sure (Y/N)?
```

なお、「del」と同等の機能を持つコマンドとして「erase」があります。

DIR

機 能

ディスクに記録されているファイルについての情報を表示します。

書 式

DIR [<ファイルスペック>][/P][/W]
DIR[/P][/W] [<ファイルスペック>]

解 説

「dir」コマンドはディレクトリに記録されているファイルの名前、大きさ、最後に修正された日付、時刻および表示したファイル数、ディスクの残り容量などの情報をディスプレイに表示します。
「dir」コマンドでは「/P」と「/W」の2つのスイッチが使用できます。

/P　　ページモードを意味し、ディスプレイいっぱいに表示されたところで表示を中断します。表示を再開するには、任意のキーを押します。
/W　　ワイドディスプレイモードを意味し、ファイル名のみを1行に表示できるだけ表示します。1行の表示幅は「mode」コマンドで設定した値によります。

「dir」コマンドにパラメータがないとき、デフォルトドライブ上にあるディスクのディレクトリの内容を表示します。デフォルトドライブAにあるディスクのディレクトリの内容を表示させるには、次のように入力します。

```
A>dir
```

パラメータとしてドライブ指定のみがあるとき、指定されたドライブにあるディスクのディレクトリの内容を表示します。ドライブBにあるディスクのディレクトリの内容を表示させるには、次のように入力します。

```
A>dir b:
```

パラメータとして拡張子なしのファイル名だけがあるとき（ドライブ指定はあってもよい)、そのドライブ上にあるディスクのディレクトリを検索して、指定されたファイル名を持つすべてのファイルの情報を表示します。次の例では、ドライブBにあり、ファイル名に「TEST1」を持つすべてのファイルの情報を表示します。

```
A>dir b:test1
TEST1     ASM     ........................
TEST1     REL     ........................
```

```
TEST1      COM     ........................
         3 files   ........................
A>
```

完全な<ファイルスペック>を指定したとき、指定されたドライブにあるディスクのディレクトリからそのファイルを検索し、その情報を表示します。

```
A>dir b:test1.asm
TEST1      ASM     ........................
         1 file    ........................
A>
```

ファイル名のパラメータとしてワイルドカード文字を使用することができます。ワイルドカード文字については、「2.2.1 ファイルの概要」を参照して下さい。なお、「dir」コマンドの呼び出しで同等の動作をするものを以下に上げます。

コマンド	同等のコマンド形式
dir	dir *.*
dir file	dir file.*
dir .ext	dir *.ext
dir .	dir *.

注 意

表示幅が36桁未満のときは、情報の一部が表示されないことがあります。このようなときは「mode」コマンドを用いて、表示幅を36桁以上に設定して下さい。

ERASE

機 能

DELコマンドと同じです。DELコマンドを参照して下さい。

FORMAT

機 能

新しいディスクを使用できるように初期化します。

書　式

FORMAT

解　説

「format」コマンドの処理はメッセージに従って対話方式で進められます。標準的なメッセージは以下のとおりです。

■ドライブが 1DD タイプのとき

```
A>format
Drive name? (A, B)          ←フォーマットするディスクが入っているドライブ名
                              を入力
Strike a key when ready     ←どれかキーを押す
Format complete
A>
```

■ドライブが 2DD タイプのとき

```
A>format
Drive name? (A, B)          ←ドライブ名を入力
1 - 1 side, double track
2 - 2 sides,double track
?                           ←数字を入力する（標準フォーマットはこの例では 1）
Strike a key when ready     ←どれかキーを押す
Format complete
A>
```

対話を中断するときは、CTRL + C を入力します。このとき、ディスプレイには次のようなメッセージが表示されます。

```
Aborted                     ←中止した
```

注　意

MSX-DOS で使用できるディスクのフォーマット（様式）を表 3.2 にあげます。

表 3.2　MSX-DOS で使用できるディスクのフォーマット

フォーマット	1DD	2DD	1DD	2DD
記録可能な総ファイル数	112	112	112	112
FAT が占めるセクタ数	2	3	1	2
1 トラックのセクタ数	9	9	8	8
サイド数	1	2	1	2
サイドあたりのトラック数	80	80	80	80
1 セクタのバイト数	512	512	512	512
使用可能なセクタ数	708	1426	630	1268

フォーマット	1DD	2DD	1DD	2DD
使用可能なセクタ数	708	1426	630	1268
使用可能な総バイト数	362496	730112	322560	649216
物理セクタ番号	1〜9	1〜9	1〜8	1〜8
トラック番号	0〜79	0〜79	0〜79	0〜79
サイド番号	0	0、1	0	0、1

いったんフォーマットが始まると中断することはできません。フォーマット中にエラーが発生すると、その内容によって次のようなメッセージが表示されます。

```
Write protected
Not ready
Disk error
```

エラーの内容は、「2.7 MSX-DOS メッセージ一覧」を参照して下さい。

3.5インチ型のMSX用ディスクドライブ装置では、全ての機種で片面80トラック(1DD)フォーマットを読み書きできるようになっています。他の装置で使う可能性のあるディスクをフォーマットするときは、このフォーマットを選ぶことが望ましいでしょう。

MODE

機 能

ディスプレイの表示文字幅を設定します。

書 式

MODE <表示文字幅>

解 説

設定できる表示文字幅は、MSX1では1〜40です。41以上の値を設定するとエラーになります。MSX2、MSX2+のときは1〜80です。設定した値が1〜32のときはスクリーンモード1(BASICのSCREEN 1)に、33〜40(MSX2、MSX2+では33〜80)のときはスクリーンモード0(BASICのSCREEN 0)に設定されます。

「mode」コマンドを実行するとディスプレイは消去されます。

PAUSE

機能

バッチコマンドの実行を一時停止します。

書式

PAUSE ［<コメント>］

解説

バッチコマンドの実行を一時停止させるときに使います。バッチファイルにこのコマンドを書き込んでおくと、バッチコマンドの実行が「pause」のところで一時停止します。このときディスプレイには以下のようなメッセージが表示されます。

Strike any key when ready
(準備ができたらどれかキーを押して下さい)

バッチコマンドの実行は、CTRL + C 以外の任意のキーを押すと再開されます。CTRL + C を入力すると、続いて次のようなメッセージが表示されます。

Terminate batch file (Y/N)?
(バッチ処理を中止しますか?)

ここで「Y」を入力するとバッチコマンドの実行は中止され、MSX-DOSのコマンド入力待ちに戻ります。「N」を入力したときはバッチコマンドの実行が再開されます。
「pause」コマンドのパラメータにコメントを与えると、それをディスプレイに表示させることができます。例えば、下の例のように、使用者に指示を与えることもできます。

```
A>newdisk
A>rem This is NEWDISK.BAT
A>pause Insert disk in drive B:
Strike a key when ready
A>format
Drive name? (A, B) b
Strike a key when ready
Format complete

A>
```

REM

機　能

何もしません。ただし結果として、「rem」のあとに続くパラメータがコメントとしてディスプレイに表示されます。

書　式

REM ［<コメント>］

解　説

「rem」コマンドは上記の動作以外に、他に何の影響も与えません。「pause」コマンドの例を参照して下さい。

REN

機　能

第1パラメータで指定したファイル名1を、第2パラメータのファイル名2に変更します。

書　式

REN <ファイル名1> <ファイル名2>
または
RENAME <ファイル名1> <ファイル名2>

解　説

デフォルトドライブ以外のファイル名を変更するときは、ファイル名1を指定する際にドライブ指定が必要です。「ren」コマンドはファイル名を変更するものなので、ファイル名2でドライブを指定しても意味を持ちません。また「ren」コマンドで他のディスクにファイルを移動させることはできません。
「ren」コマンドではワイルドカード文字を使用することができます。このとき、ファイル名1で指定したファイルとファイル名2で指定したファイルとの間で、各文字が1対1に対応して処理されます。例えば次の例では、「.LST」拡張子を持つ全てのファイルの拡張子を「.PRN」に変えます。

```
A>ren *.lst  *.prn
```

次の例では、ドライブB上のファイルABCDEをADCBEという名前に変更することになります。

```
A>ren b:abcde ?d?b?
```

なお、「rename」は「ren」と同等のコマンドです。

RENAME

機 能

RENコマンドと同じです。RENコマンドを参照して下さい。

TIME

機 能

時刻の表示、変更を行います。

書 式

TIME [<時>[:<分>[:<秒>]]][{A | P}]

解 説

パラメータなしで入力したときは次のように表示されます。

```
A>time
Current time is hh-mm-ss.tt {a | p}
Enter new time:
```

hh、mm、ss.tt、aまたはpは内蔵の時計が示す時刻です。

hh	1～12の数字で時を示します。
mm	0～59の数字で分を示します。
ss.tt	ssは0～59の数字で秒を示します。また、ttは1/100秒を示しますが、表示される数字は「00」です。
aまたはp	午前(a)か午後(p)かを示します。

表示された時刻を変更する必要がないときはここでリターンキーを押します。
パラメータとして時刻を入力すると、メッセージは表示されずにそのまま時刻の変更が行われます。

時刻の入力には数字を使います。「.tt」を入力する必要はありません。また、最後の数字の後に「a」または「p」をつけて、午前（am）と午後（pm）を指定することができます。「a」や「p」をつけないと、24時間制で時刻を指定したとみなされます。したがって、次の2つの例では、最初のものは午前10時10分30秒に変更し、次のものは時刻を午後10時10分30秒に変更します。

```
A>time 10:10:30
A>time 10:10:30p
```

時刻の入力に使用できるパラメータの範囲は次のとおりです。

時	0〜23	24時間制で入力したと見なされる
	1〜12 {a | p}	aまたはpを省略すると午前と見なされる
分	0〜59	
秒	0〜59	

時：分：秒の入力は、コロン（：）で区切ります。パラメータや区切り記号が誤っていると、次のメッセージが表示されるので、もう一度正しく入力して下さい。

```
Invalid time          ←時刻の指定が違います
Enter new time:       ←時刻を入力して下さい
```

TYPE

機 能

<ファイルスペック>で指定したファイルの内容をディスプレイに表示します。

書 式

TYPE <ファイルスペック>

解 説

「type」コマンドはアスキーファイルの内容を表示させるために用います。<ファイルスペック>の指定にはワイルドカード文字を使用できますが、ディレクトリを捜して最初に指定に該当したファイルの内容のみを表示します。
バイナリファイルの内容を表示させると、ベルコード、フォームフィードコード、およびエスケープシーケンスを含むコントロールシーケンスがディスプレイに送られ、正しい表示が行われません。
表示させるファイルの名前を捜すには「dir」コマンドを使います。

VERIFY

機　能

ディスクに書き込む際に、読み込みチェックをするかどうかを設定します。

書　式

VERIFY {ON | OFF}

解　説

「verify」コマンドは、書き込みに際してファイルが正しく書き込まれたことを読み出してチェックするかするかどうかをかを設定します。「verify」を「on」にすると、ディスクに書き込むごとにチェックを行い、「off」にするとチェックは行いません。
「verify on」のときは書き込みに時間がかかるので、通常は「off」にします。MSX-DOSの起動時には、「verify off」に設定されています。重要なファイルをコピーするとき、例えばマスターディスクの予備を作るときなどには、「verify on」にすると良いでしょう。
ベリファイ機能はオプションであり、ドライブによってはその機能がないものもあります。

2.5 特殊キーの機能

2.5.1 特殊キーとは

MSX-DOSで使用するキーには、文字キーの他に特別な機能を持つ「特殊キー」があります。これらのキーを使うことで、MSX-DOSの操作がより簡単になります。

特殊キーには、[CTRL]、[DEL]、[HOME]、[INS]、[SELECT]などがあります。特殊キーのほとんどは単独で使用されますが、[CTRL]キーだけは、それを押しながら他の文字キーを押して使います。

注 意

[CTRL]+文字は、[CTRL]キーを押しながら文字のキーを押すことを表します。

2.5.2 コマンド行の編集機能

1. 編集機能

MSX-DOSには、直前に入力されたコマンド行を記憶し、それを再び実行したり、修正して実行する機能があります。この機能を利用すれば、同じようなコマンドを繰り返し入力する手間を減らすことができます。この記憶されたコマンド行を「テンプレート」といいます。

図3.7は入力されたコマンド行がどのように記憶され、呼び出されるかを表したものです。

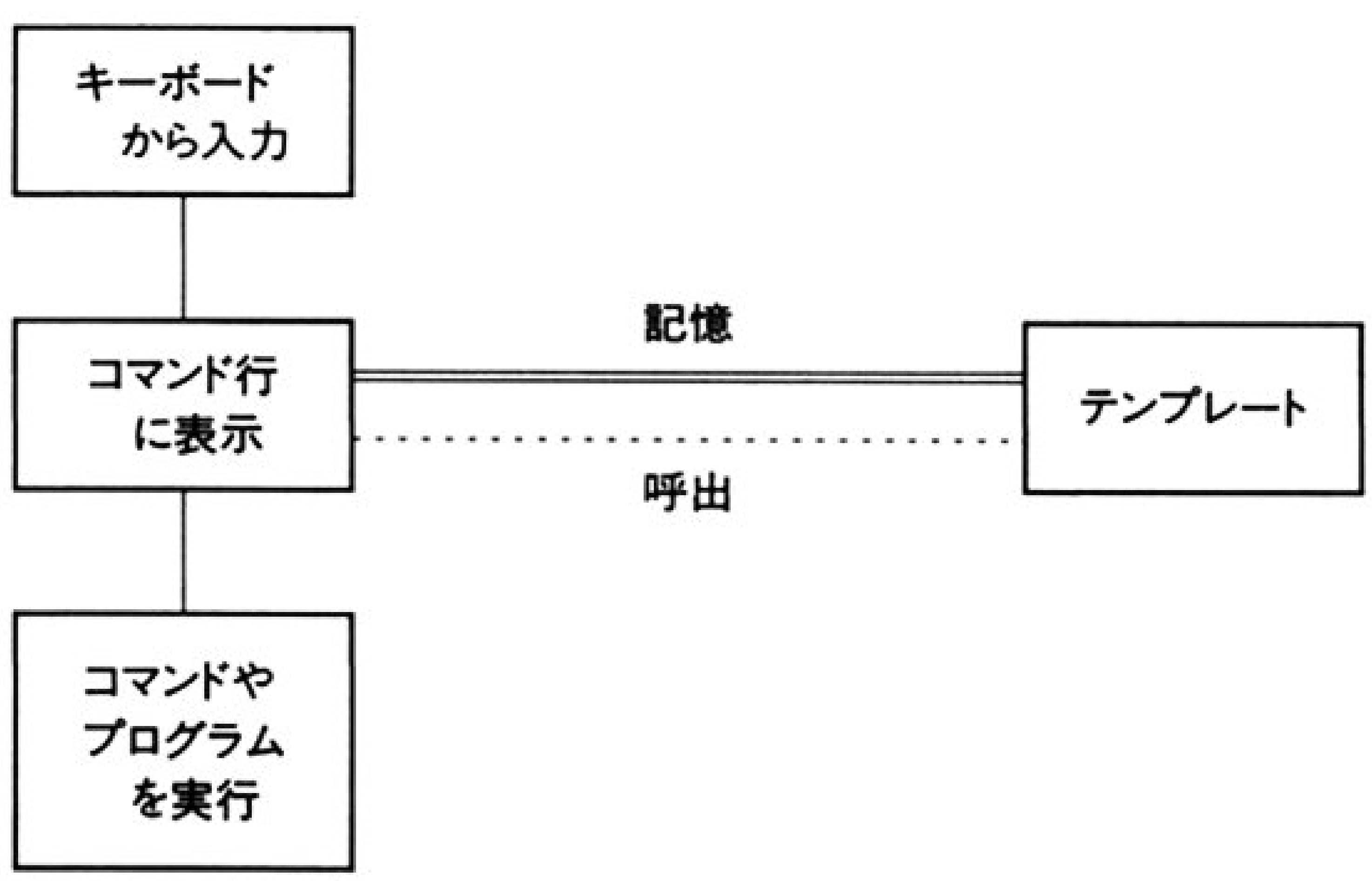


図3.7 コマンド行とテンプレート

コマンド行の編集に使われる特殊キーの機能を表3.3に示します。1つの機能に複数のキーが割り当てられているときは、どのキーを用いても同じ効果があります。

表3.3　特殊キーの機能

特殊キー	機　能	内　容
[→] [CTRL]+[¥]	1字コピー	テンプレートからコマンド行へ1文字コピーします。
[SELECT] [CTRL]+[X]	指定コピー	テンプレートからコマンド行へ、指定された文字の直前までの文字をコピーします。
[↓] [CTRL]+[_]	1行コピー	テンプレート内に残っている全ての文字をコマンド行へコピーします。
[DEL]	1字スキップ	テンプレート内の1文字を飛び越します(コピーしません)。
[CLS] [CTRL]+[L]	指定スキップ	テンプレート内の文字を指定された文字の直前まで飛び越します（コピーしません)。
[↑] [ESC] [CTRL]+[^] [CTRL]+[[] [CTRL]+[U]	取消	現在のコマンド行を取り消します。テンプレートの内容は変更されません。
[INS] [CTRL]+[R]	挿入	挿入モードに入ったり抜け出たりします。
[←] [BS] [CTRL]+[H] [CTRL]+[]]	後退	コマンド行から最後の1文字を除去します。
[HOME] [CTRL]+[K]	新しい行を作る	コマンド行をテンプレートにコピーします（新しいテンプレートを作ります)。
[CTRL]+[J]	行換え	ディスプレイの表示のみ改行します。

2. 編集機能の使用例

ここで実際に例を上げてコマンド行の編集機能を解説します。なお、テンプレートに関するキー操作はディスプレイには表示されません。ここではわかりやすくするために、特殊キーを《 》で囲んで示します。

以下のコマンド行は、ファイル「PROG.COM」のファイル情報を表示するものです。このコマンドを入力すると「dir」コマンドの実行とともに、コマンド行がテンプレートに記憶されます。

A>dir prog.com

もう一度このコマンド行を繰り返すには、↓とリターンキーを使います。↓を押すと、テンプレートがコマンド行にコピーされます。そして、リターンキーを押すとコマンドとして実行されます。

A>《↓》dir prog.com

次に、ファイル「PROG.MAC」のファイル情報を表示し、続けてその内容を表示しましょう。

現在、テンプレートには「dir prog.com」と記憶されています。まず、SELECT を押し、続いて C と押すと、「c」の直前までの文字がコマンド行にコピーされます。これに続けて「mac」と押すと、目的のコマンド行ができあがります。

A>《SELECT》《C》dir　prog.
A>dir prog.mac

ここでリターンキーを押すと、このコマンド行が実行され、今度は「dir prog.mac」がテンプレートに記憶されます。では「type prog.mac」というコマンド行を作ってみます。それには次のようにキーを押します。

A>type《INS》　《↓》

新しく入力された文字は直接コマンド行に入力されます。「dir」とそれに続くスペースの4文字が「type」に変換され、そしてコマンドとパラメータの間に必要なスペースを入れるために INS とスペースを入力します。そして、最後にテンプレートの残りの「prog.mac」をコマンド行にコピーします。すると、コマンド行は次のようになります。

A>type prog.mac

次に、コマンド行の編集機能を用いてテンプレートを修正する手順を説明します。ここでは、「type」とするべきところを「byte」と入力してしまったとします。これを修正するには HOME キーを使います。HOME には現在のコマンド行をテンプレートにコピーし、新しいテンプレートを作る機能があります。HOME を押すと新しいテンプレートを作った目印として、コマンド行の末尾に「@」が表示され、カーソルは次の行に移ります。

A>byte prog.mac《HOME》@

ここで、→ と ↓ を使いテンプレートの内容を修正しながらテンプレートをコマンド行にコ

ピーします。[→] はテンプレートから1文字をコマンド行にコピーする機能を持っています。間違えた「b」を「t」に、さらに「t」を「p」に置き換え、残りを↓でコピーします。

```
A>byte prog.mac@
t《→》yp《↓》e prog.mac
```

また、同じことは以下のように [DEL] キーと [INS] キーを用いても可能です。

```
A>byte prog.mac@
《DEL》《DEL》《→》t《INS》yp《↓》e prog.mac
```

これを分解すると次のようになります。

タイプ	コマンド行	意　味
《DEL》		1文字目をスキップ
《DEL》		2文字目をスキップ
《→》	t	3文字目をコピー
《INS》yp	typ	ypを挿入
《↓》	type prog.mac	残りをコピー

2.5.3 その他の機能

MSX-DOSにはコマンド行の編集に用いる特殊キーの他に、表3.4のような特殊キーがあります。

表3.4　その他の特殊キー

機　能	特殊キー	内　容
中止	[CTRL]＋[C]	現在動作中のコマンドやプログラムの実行を中止します。
印刷	[CTRL]＋[P]	ディスプレイへの表示をプリンタへも出力します。
印刷取消	[CTRL]＋[N]	「[CTRL]＋[P]」の動作を取消します。
表示停止	[CTRL]＋[S]	ディスプレイへの表示を一時停止します。任意のキーを押すと表示は再開します。

2.6 バッチ処理

2.6.1 バッチファイルの作成

バッチコマンドを利用するには、まずディスク上にバッチファイルを作成しなければなりません。バッチファイルは、「copy」コマンドを用いて次のような手順で作成します。バッチファイルはエディタでも作成できます。

■キーボードからの入力を「AUTOEXEC.BAT」というファイル名でドライブAのディスクにコピーする

```
A>copy con autoexec.bat
rem This is AUTO EXECUTE FILE          この入力がファイルに記録される
basic funckeys.ini
^Z                                     ←ファイルの終りには、必ず[CTRL]+[Z]を入力
                                         する
          1  file copied
A>
```

「^Z」はファイルがそこで終ることを示す記号で、エンドオブファイルマーク（EOF）といい、ファイルの最後には必ず書き込まれなければなりません。

上の例の「AUTOEXEC.BAT」がディスク上に作成されたかどうか、「type」コマンドで表示させると以下のようになります。

```
A>type autoexec.bat
rem This is AUTO EXECUTE FILE
basic funckeys.ini

A>
```

2.6.2 パラメータの使用例

バッチファイルでは、「%」のあとに1桁の数字を用いた仮パラメータを使用することができます。この仮パラメータは、バッチコマンド実行時にコマンド行に書かれたファイルスペックやスイッチ、パラメータと置き換えて処理されます。これを利用すれば、1つのバッチファイルで様々なファイルに対する処理を行うことができます。

仮パラメータは、%0から%9までの10個が使用できます。%0はコマンド名自身と置き換えら

れ、%1から%9までの9個が実パラメータと置き換えられます。次の例では、仮パラメータを使用したバッチファイルで、アセンブラを用いてコマンドファイルを作っています。

```
A>type asm.bat
rem  This is ASM.BAT
rem  START BATCH FILE
dir
m80=%1.asm
m80=%2.asm
l80  %1, %2, %1/n/e
dir

A>
```

このバッチファイルを使って、test1.asmとtest2.asmをアセンブルし、コマンドファイルを作成します。

```
A>asm test1 test2
A>rem This is ASM.BAT
A>rem START BATCH FILE
A>dir
MSXDOS      SYS      2432  85-08-23   9:29p
COMMAND     COM      6656  85-09-02  10:10p
M80         COM     20224  85-04-01  10:27p
L80         COM     10725  85-04-01  10:25p
ASM         BAT       117  85-07-17  12:24p
TEST1       ASM       256  85-07-18  12:20a
TEST2       ASM       256  85-07-18  12:23a
          7 files  317440 bytes free
A>m80 = test1.asm

No Fatal error(s)

A>m80 = test2.asm

No Fatal error(s)

A>l80 test1,test2,test1/n/e

MSX.L-80   1.00 01-Apr-85     (C)   1981, 1985 Microsoft

Data     0100    0141     <    65>

43504 Bytes Free
[0000    0141         1]

A>dir
MSXDOS      SYS      2432  85-08-23   9:29p
COMMAND     COM      6656  85-09-02  10:10p
M80         COM     20224  85-04-01  10:27p
L80         COM     10725  85-04-01  10:25p
ASM         BAT       117  85-07-17  11:24p
TEST1       ASM       256  85-07-18  12:20a
TEST2       ASM       256  85-07-18  12:23a
TEST1       REL       128  85-07-18  12:24a
```

```
TEST2        REL        128  85-07-18  12:25a
TEST1        COM        128  85-07-18  12:25a
          10 files     314368 bytes free
A>
A>
```

このように、%1、%2という2つの仮パラメータが、コマンド行から入力した順にtest1、test2と置き換って処理されます。

2.6.3 MSX-DOS起動時の自動実行機能

「AUTOEXEC.BAT」というバッチファイル名は、MSX-DOSが使用します。「AUTOEXEC」とは自動実行(AUTO EXECution)という意味で、MSX-DOS起動時に自動的に実行されるバッチファイルです。

MSX-DOSが起動すると、MSXはドライブAに入っているディスクから「AUTOEXEC.BAT」というバッチファイルを捜し、見つかればバッチコマンドとして実行します。「AUTOEXEC.BAT」がないときは通常どおりに起動します。

なお、時計機能がないMSXで、「AUTOEXEC.BAT」が実行されると、起動時に必要な日付を入力する手続は省略されますがMSX-DOSの動作に影響はありません。

2.7 MSX-DOSメッセージ一覧

MSX-DOSは、動作中にいろいろなメッセージを表示します。このメッセージには、使用者に対してなんらかの処置を求める入力待ちメッセージと、エラーが起こったことを伝えるエラーメッセージとがあります。

解説はメッセージのアルファベット順に、次のような構成で行います。なお、ディスプレイには対処法が表示されないので、そのメッセージに対する対処についても解説を加えます。

表示されるメッセージ

コマンド

そのメッセージが表示される可能性があるコマンドなど。

意　味

メッセージの意味の説明。

内　容

メッセージが表示された原因、内容などの説明。

対処方法

そのメッセージへの対処方法。

なお、コマンドの項に「COMMAND」とあるものは、そのメッセージが「COMMAND.COM」によって管理される外部コマンドやバッチコマンドにおいて表示される可能性があることを示します。

2.7.1 入力待ちメッセージ

Are you sure(Y/N)?

コマンド

DEL

意　味

よろしいですか <Y/N>?

内　容

「del」コマンドでファイル名の指定に「*.*」を用いると（全部のファイルを削除という意味）表示されます。

対処方法

全ファイルを削除してもよいのなら「Y」を押します。「N」を押すと、削除せずにコマンド入力待ちに戻ります。

Boot error
Press any key for retry

コマンド

MSX-DOS 起動時

意　味

MSX-DOS を起動できません。
任意のキーを押し、もう一度起動して下さい。

内　容

ドライブ A に挿入されているディスクに「MSXDOS.SYS」が入っていないため、MSX-DOS を起動することができません。

対処方法

「MSXDOS.SYS」が記録されているディスクをドライブ A に挿入したのち、任意のキーを押してもう一度起動して下さい。

Disk error reading drive <ドライブ番号>
Abort, Retry, Ignore?
Disk error writing drive <ドライブ番号>
Abort, Retry, Ignore?

コマンド

ディスクアクセス中（COMMAND、COPY、DEL、DIR、REN、TYPE）

意　味

ディスクアクセス中にエラーが発生しました。
中止＜A＞、再試行＜R＞、無視＜I＞?

内　容

ディスクにキズがあるなどの物理的な事故があったとき、またはドライブに故障があるときにこのエラーが発生します。

対処方法

何度か「R」を入力し、それでもエラーが繰り返されるようであれば、そのディスクは使用不能です。「A」を入力してコマンドを中止してから、新しいディスクでその処理を行って下さい。それでもエラーが発生したらドライブの故障と考えられます。

Insert disk with batch file and strike any key when ready

コマンド

バッチコマンド

意　味

バッチファイルのあるディスクをドライブに入れて下さい。
準備ができたら任意のキーを押して下さい。

内　容

バッチコマンド実行中に、そのバッチファイルがあるディスクを引き抜いてしまったときに表示されます。

対処方法

実行中のバッチファイルがあるディスクをもとのドライブに挿入してから任意のキーを押します。

Insert diskette for drive <ドライブ番号> and strike a key when ready

コマンド

COMMAND、COPY、DEL、DIR、FORMAT、REN、TYPE

意　味

ディスクをドライブ<ドライブ番号>に入れて下さい。
準備ができたら任意のキーを押して下さい。

内 容

仮想ドライブ機能が動作しているとき、複数のドライブに対して処理を行うコマンドを実行すると表示されます。

対処方法

「2.1.4 3.仮想ドライブ機能」を参照して下さい。

Insert DOS disk in default drive and strike any key when ready

コマンド

すべてのコマンド

意 味

COMMAND.COMのあるディスクをデフォルトドライブに入れて下さい。
準備ができたら任意のキーを押して下さい。

内 容

外部コマンド終了時に、デフォルトドライブに挿入されているディスクに「COMMAND.COM」が存在しないと、このメッセージが表示されることがあります。
また、MSX-DOS起動時に、ドライブAに挿入されているディスクに「MSXDOS.SYS」だけがあり、「COMMAND.COM」がないときにもこのメッセージが表示されます。

対処方法

「COMMAND.COM」が入ったディスクをデフォルトドライブに挿入してから任意のキーを押します。

Invalid date Enter new date

コマンド

DATE

意　味

日付の指定が違います。

新しい日付を入力して下さい。

内　容

日付の指定に許されていない値を入力すると表示されます。

対処方法

正しい値をもう一度入力します。

Invalid time
Enter new time

コマンド

TIME

意　味

時刻の指定が違います。

新しい時刻を入力して下さい。

内　容

時刻の指定に許されていない値や区切り記号を入力すると表示されます。

対処方法

正しい値をもう一度入力します。

Not ready error reading drive <ドライブ番号>
Abort, Retry, Ignore?

コマンド

COMMAND、COPY、DEL、DIR、REN、TYPE

意　味

ドライブ＜ドライブ番号＞は読み込む準備ができていません。
中止＜A＞、再試行＜R＞、無視＜I＞?

内　容

ディスクが正しくドライブに挿入されていないときや、ドライブに故障があったときに表示されます。

対処方法

ディスクが正しくドライブに挿入されているかどうかを確認し、「R」を押します。

Not ready error writing drive <ドライブ番号> Abort, Retry, Ignore?

コマンド

COMMAND、COPY、DEL、REN

意　味

ドライブ＜ドライブ番号＞に書き込む準備ができていません。
中止＜A＞、再試行＜R＞、無視＜I＞?

内　容

ディスクが正しくドライブに挿入されていないときや、ドライブに故障があったときに表示されます。

対処方法

ディスクを引き抜いてしまったときは正しいディスクを挿入し「R」を入力します。
ドライブの故障のときは修理して下さい。

Strike a key when ready

コマンド

DIR/P、FORMAT、PAUSE

意　味

準備ができたら任意のキーを押して下さい。

内　容

一時停止した画面の表示またはバッチコマンドの実行を再開するかどうか指示を得なければならないときに表示されます。

「format」コマンドでは、ディスクの準備のために表示されます。

対処方法

必要な処置をとってから任意のキーを押します。

Terminate batch file (Y/N)?

コマンド

バッチコマンド

意　味

バッチ処理を中止しますか（Y/N）？

内　容

バッチ処理中に CTRL ＋ C を押すと表示されます。

対処方法

バッチコマンドを中止するときは「Y」を、続行するときは「N」を入力します。「N」を入力押したとき、バッチコマンドは次のコマンド行から続行されます。

Write protect error writing drive <ドライブ番号> Abort, Retry, Ignore?

コマンド

COMMAND、COPY、DEL、REN

意　味

ドライブ＜ドライブ番号＞に対して書き込みができません。

中止＜A＞、再試行＜R＞、無視＜I＞?

内　容

書き込みの対象となっているディスクが書き込み禁止になっている（ライトプロテクトノッチが書き込み不可側にセットされている）ときに発生します。

対処方法

ライトプロテクトノッチを書き込み可にセットしなおしてから「R」を入力します。

ファイルのコピー中などに、ディスクを入れ間違えたときは、「A」を入力して処理を中止し、もう一度最初から処理を行います。

注　意

「R」を入力するときには、ディスクを入れ替えないで下さい。ディスクを入れ替えると、そのディスクが壊れ、データも正しく書き込まれません。

以下のメッセージも入力待ちメッセージですが、ここでは解説を省略します。「2.4 コマンド一覧」のそれぞれのコマンドの項を参照して下さい。

Current date is www yy-mm-dd
Enter new date:

コマンド

DATE

Current time is hh:mm:ss.tt {a | p}
Enter new time:

コマンド

TIME

2.7.2 エラーメッセージ

Bad command or file name

コマンド

COMMAND

意　味

コマンドまたはファイル名が違います。

内　容

入力した外部コマンド名またはバッチファイル名が指定したドライブ上にないときや、コマンド名、ファイル名の入力を間違えたときに発生します。

対処方法

- ドライブ指定で、そのファイルが記録されているディスクが挿入されているドライブを指定します。
- そのファイルが記録されているディスクをドライブに挿入し、そのドライブ名を指定します。
- 正しいファイル名を入力します。

Bad FAT, drive <ドライブ番号>

コマンド

COMMAND、COPY、DEL、DIR、REN、TYPE

意　味

FAT の内容が破壊されています。

内　容

FAT の内容に異常があったときに発生します。

対処方法

致命的なエラーで、そのディスクに記録されているファイルの内容は保証できません。

このエラーが出たときは、読み出せるファイルがあれば(「type」コマンド、「copy」コマンドを使って確認する)、別のディスクにコピーして救済して下さい。エラーが起こったディスクはフォーマットしなおす必要があります。

Bad parameter

コマンド

FORMAT

意　味

パラメータが違います。

内　容

「format」コマンドがフォーマットの様式を選択するようになっていたときに、間違った値を入力するとこのエラーが発生します。

対処方法

「format」コマンドをもう一度起動し、正しい値を入力して下さい。

Disk error

コマンド

FORMAT

意　味

ディスクにエラーが発生しました。

内　容

「format」コマンド実行中にディスク入出力関係のなんらかのエラーが発生しました。

対処方法

他のコマンドを実行して具体的にどのようなエラーなのかを調べ、それに対処します。

File cannot be copied onto itself

コマンド

COPY

意　味

ファイルはそれ自身にはコピーできません。

内　容

ファイルをそれ自身が記録されているディスクにコピーしようとすると、このエラーが発生します。例えば、次のようなコマンドを実行したときなどです。

```
A>copy file.ext
A>copy file.ext a:
A>copy *.* a:
```

対処方法

■同じディスクにコピーしたいときは違うファイル名でコピーします。
■違うディスクにコピーします。

File creation error

コマンド

COPY

意　味

ファイルを作れません。

内　容

すでに記録可能な総ファイル数まで使用されているディスク上に、新しくファイルを作成しようとするとこのエラーが発生します。

対処方法

新しいディスクを使用して下さい。

File not found

コマンド

DIR、TYPE、DEL、COPY

意　味

ファイルが見つかりません。

内　容

<ファイルスペック>で指定したファイルが、指定したドライブに存在しないときや、ファイル名の入力を間違えたときに発生します。

対処方法

- そのファイルが記録されているドライブを指定します。
- そのファイルが記録されているディスクをドライブに入れます。
- 正しいファイル名を入力します。

Insufficient disk space

コマンド

COPY

意　味

ディスクの容量が足りません。

内　容

ファイルを作ったりコピーするために充分な容量がディスクに残っていないときに発生します。

対処方法

新しいディスクまたは充分に空きのあるデイスクを用いてその処理を行います。

Invalid drive specification

コマンド

COMMAND、DIR、TYPE、REN、DEL、COPY

意　味

ドライブの指定が違います。

内　容

ドライブ指定で、使用することができないドライブを指定したときに発生します。例えば、ドライブBまでしか使用できないのにドライブCを指定したときなどです。

対処方法

正しいドライブ指定を行います。

Invalid parameter

コマンド

MODE、VERIFY

意　味

パラメータが違います。

内　容

「mode」コマンドで許されている値の範囲を越えて指定したとき、「verify」コマンドでは「on」、「off」以外のものを指定したときに発生します。

対処方法

正しいパラメータを入力します。

No enough memory

コマンド

FORMAT、MSX-DOS起動時

意　味

メモリが足りません。

内　容

MSX-DOSが起動するために必要なメインRAM (64KB) がないときや、「format」コマンドを実行するために充分なメモリがないときに発生します。

対処方法

ドライブの台数が多すぎることが考えられます。そのときは、CTRL キーを押しながら立ちあげて、2ドライブシミュレータ機能を切り離すなどの方法でドライブ数を少なくします。

Not ready

コマンド

FORMAT

意　味

ディスクが入っていません。

内　容

フォーマットの対象となっているドライブにディスクが入っていないときに表示されます。

対処方法

ドライブにディスクを挿入します。

Program too big to fit in memory

コマンド

COMMAND

意　味

プログラムが大き過ぎて、メモリに入りません。

内　容

使用できるメモリより大きいプログラムを実行しようとしたときに発生します。

対処方法

- 接続しているディスクの数を減らして、使用できるメモリを増やします。
- 大きさが小さくなるように、プログラムを変更します。

Rename error

コマンド

REN

意　味

このファイル名変更は間違っています。

内　容

ファイル名を変更する際に、新しいファイル名としてすでに存在するファイル名を指定したときに発生します。

対処方法

dir コマンドでどのようなファイルが記録されているかを確認し、そのディスクにはないファイル名を指定します。

Unsupported media type error reading drive <ドライブ番号>

コマンド

ディスクアクセス中

意　味

<ドライブ番号>のディスクはこのドライブでは扱えません。

内　容

そのドライブで扱うことができない種類のディスクに対して入出力しようとしたときに発生します。このとき自動的に Abort します。

対処方法

そのドライブで扱えるタイプのディスク以外は使用しないで下さい。ただし、そのディスクがフォーマットされていない可能性もあります。

Write error

コマンド

COPY

意　味

書き込みができません。

内　容

何らかの理由で出力先のファイルに書き込めなくなったときに発生します。

対処方法

もう一度最初からやり直します。

Write protected

コマンド

FORMAT

意　味

ディスクが書き込み保護されています。

内　容

フォーマットの対象となっているディスクが書き込み禁止になっている（ライトプロテクトノッチが書き込み不可側にセットされている）ときに発生します。

対処方法

ライトプロテクトノッチを書き込み可にセットしなおしてから、もう一度「format」コマンドを実行します。

3章 MSX-DOSの構造

3.1 MSX-DOSの起動

MSX-DOSは以下の手順で起動します。

1. MSXをリセットすると、始めにすべてのスロットを調べ、調べたスロットの先頭に「41H、42H」の2バイトが書き込まれていれば、そのスロットには何らかのROMプログラムが接続されていると判断し、ROMのヘッダ部分にアドレスを設定されたINIT(初期化)ルーチンを実行します。ディスクインターフェイスROMのINITルーチンの場合は、まずそのインターフェイスに接続されているドライブのために、ワークエリアを確保します。「第2部 1章　ブートシーケンス」を参照して下さい。
2. すべてのスロットを調べ終わったら、【H.STKE(FEDAH)】を参照します。このアドレスの内容がC9Hでなければ、Disk BASICの環境を設定し、H.STKEにジャンプします。
3. もし、上記の調査で、H.STKEの内容がC9Hであれば、TEXTエントリを持つカートリッジを各スロットで捜し、あればDisk BASICの環境を設定した後、そのカートリッジのBASICプログラムを実行します。
4. ブートセクタ（論理セクタ0）の内容をC000H〜C0FFHへ転送します。このとき、「DRIVE NOT READY」か「READ ERROR」が発生したり、転送されたセクタの先頭が「FBH」か「E9H」ではなかったときは、Disk BASICが起動します。
5. C01EHに転送されたルーチンがCYフラグをリセットした状態でコールされます。通常は、このアドレスには「RET NC」のコードが書き込まれているため、何も実行しないでリターンします。

   このとき、スロットの状態は以下のようになっています。しかし、BIOSコールやファンクションコールによって、ページ1をRAMに切り換えることはできないので、ユーザープログラムの起動には適当ではありません。

ページ	内　容
0	BIOS ROM
1	DISK ROM
2	RAM
3	RAM

6. RAMの容量を調べます。このとき、RAMの内容は壊れません。もし、RAMが64KB未満だったら、Disk BASICが起動します。
7. MSX-DOSの環境を初期化した後、CYフラグをセットした状態でC01EHがコールされます。通常は、このアドレスには「RET NC」の後ろに標準のブートプログラムが書き込まれています。この部分に任意の機械語プログラムのブートプログラムを書き込んでおけば、そのプログラムが自動的に起動します。
このとき、スロットの状態は以下のようになっています。

ページ	内　容
0	RAM
1	DISK ROM
2	RAM
3	RAM

また、レジスタには、以下の情報が入っています。

レジスタ	内　容
A	0ならばPOWER ON直後を示す
DE	この内容をコールすると、ページ1のFDD ROMがRAMに切り換わる
HL	ディスクエラー処理ルーチンへのポインタへのポインタ（「第2部 7.9.3 エラー処理」参照）

標準のブートプログラムは、MSXDOS.SYSを100Hからのアドレスにロードし、100Hにジャンプします。MSXDOS.SYSがないときは、Disk BASICを起動します。
8. MSXDOS.SYSは自分自身を高位アドレスに転送した後、COMMAND.COMを100Hからのアドレスにロードし、その先頭へジャンプします。COMMAND.COMがないときは、「INSERT A DISKETTE」のメッセージが表示されて、正しいディスケットがドライブにセットされるのを待ちます。

9. COMMAND.COMも自分自身をMSXDOS.SYSのすぐ下の高位アドレスに転送し、COMMAND.COM自身が実行を始めます。
10. MSX-DOSが最初に起動したとき、「AUTOEXEC.BAT」という名前のファイルがあれば、それをバッチファイルとして実行します。MSX-DOSが起動せず、Disk BASICが立ち上がったときは、「AUTOEXEC.BAS」という名前のBASICプログラムがあれば、それを実行します。

3.2 プログラムの起動から終了まで

MSX-DOSは外部コマンドの形でコマンドを追加・変更できる拡張性の高いオペレーティングシステムです。ユーザーの作ったプログラムも外部コマンドとして簡単に実行することができます。

外部コマンドは、アセンブラやコンパイラなどを利用して作られた、「COM」という拡張子のついたファイルです。コマンドプロセッサは指定されたコマンド名+「COM」という名前のファイルをディスク上に見つけると、それをメモリ上にロードして実行します。このファイルの内容はメモリの100H番地からロードして、そのまま実行できるような形式になった機械語のプログラムです。

3.2.1 プログラムの起動

MSX-DOSの基本的な動作は、コマンド行を入力し実行させる、という処理の繰り返しです。この時、コマンド行の入力からコマンドの解釈実行までのユーザーインターフェイスを受け持つのがコマンドプロセッサである「COMMAND.COM」というプログラムです。入力したコマンドが外部コマンドのとき、コマンドプロセッサは、それを以下のようにして実行に移します。

1. コマンド行のパラメータについて、その長さをシステムスクラッチエリアの80H番地に、実際の文字列を81H番地以降に格納する。さらに最初の2つのパラメータをファイル名とみなし、それをFCBの形式でそれぞれシステムスクラッチエリアの5CH番地および6CH番地以降に格納する。
2. 外部コマンドを100H番地以降に読み込む。
3. 100H番地にジャンプする。

MSX-DOSでは、システムとプログラムとの間でデータなどを受け渡すために、メモリの0

〜FFH番地を使っています。これをシステムスクラッチエリアといいます。

コマンドプロセッサはまず、コマンド行のパラメータを渡すために、システムスクラッチエリアの設定を行います。外部コマンドは、システムスクラッチエリアを参照することにより、渡されたパラメータを知ることができます。このために2つの方法が使われます。

1. 5CH番地または6CH番地はそのままファイルのアクセスに使える形式になっているので、ファイル名をパラメータとするようなプログラムではこの方法が便利です。ただし、この方法では両FCB(5CH、6CH)の先頭アドレスが16バイトしか離れていないので、完全なFCBとして使用できるのはどちらか片方だけになります。FCBについては、「3.3.2 ファイルの入出力」の項で詳しく説明します。
2. 80H番地には全パラメータがそのまま入っているので、ファイル名以外をパラメータとしたり、3つ以上のファイル名を扱うようなプログラムではこの方法を使用します。

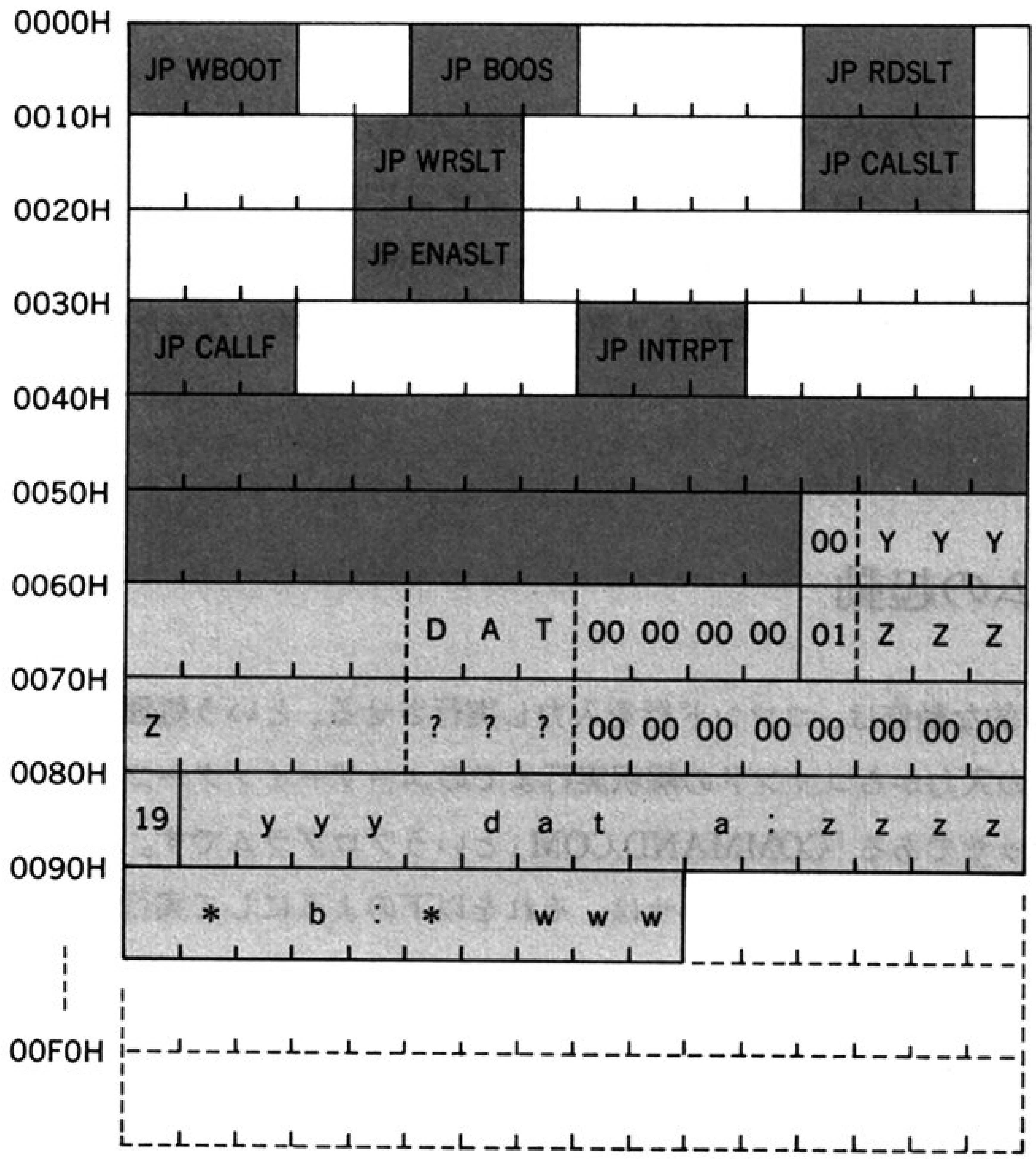


図3.8 システムスクラッチエリア

図3.8はシステムスクラッチエリアの内容です。これは、

A>test yyy.dat a:zzzz.* b:*.www

というコマンドを入力して、外部コマンド「test」を実行したときの例です。80H番地以降にはこのコマンドに渡すパラメータとその長さが入っているのがわかります。また、前述したように、5CH番地と6CH番地に最初の2つのパラメータが形式を整えられて入っています。先頭1バイトには、ドライブが無指定のときは「0」、A:であれば「1」、B:なら「2」・・・という値が入ります。また、小文字は大文字に変換され、「*」は複数の「?」に展開されます。システムスクラッチエリアの5CH～FFH番地は、このようにパラメータを渡すために使いますが、プログラムが実行を開始した後は、ワークエリアとして使うこともできます。

図3.9はプログラムが実行を開始したときのメモリマップです。00H～FFH番地はシステムスクラッチエリアで、プログラムはその直後の100H番地からロードされます。100H番地から始まり、06H～07H番地の内容で示されているアドレスの1バイト手前までの領域は、TPA (Transient Program Area =一時プログラム領域) と呼ばれ、プログラムで自由に利用してよい領域です。

システムスクラッチエリアの先頭の方には、いくつかのジャンプ命令が置かれています（図3.8参照)。プログラムはこれらの決められたアドレスをコールすることで、MSXシステムの持つ各種の機能が実行できます。00H番地にあるのがウォームスタートのためのエントリ、05H番地にあるのが4章で解説するファンクションコールのためのエントリです。この2つはMSX-DOSが持っている機能をプログラムから利用するために用意されています。またRDSLT、WRSLT、CALSLT、ENASLT、CALLFおよびINTRPTは、MSX BIOSにある同名のファンクションと同じ機能を持っています。INTRPTは割り込み時に使われますが、その他のエントリはMSX BIOSと同様に利用できます。MSX-DOS自身もこれらのエントリを使用しているので、0H～5BH番地（■の部分）を破壊するとシステムダウンにつながります。

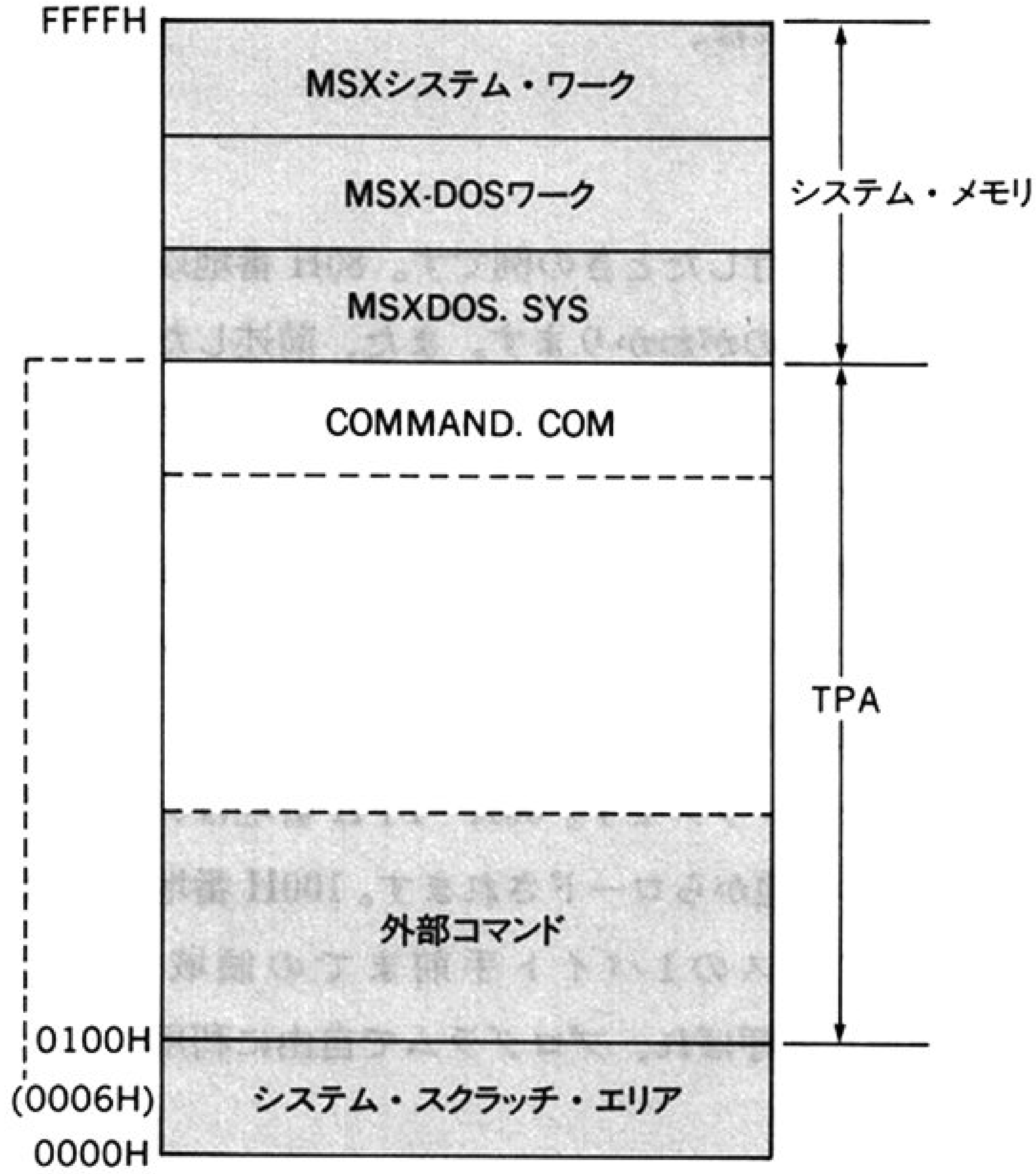


図 3.9 外部コマンド実行時のメモリマップ

05H 番地はファンクションコールのためのジャンプ命令です。06H〜07H 番地はジャンプ先のアドレスで、これが TPA の上限を示しているということは、ジャンプ先が MSXDOS.SYS の先頭になっているということです。つまり、05H〜07H 番地はジャンプ命令であると同時に TPA の上限を示すという 2 重の働きを持っています。

3.2.2 プログラムの終了

プログラムは実行が終わると、次の 3 つの方法のいずれかで MSX-DOS のコマンドレベルに戻ることができます。

1. スタックポインタを変更していない場合、「RET」を実行する。
2. 後述のファンクションコールを使って、「システムリセット」を行う。
3. 00H 番地（ウォームスタートのためのエントリ）にジャンプする。

コマンドプロセッサはスタックの先頭に 0 を置いて、外部プログラムを呼び出すので、1.は 2.と同じ結果になります。ただし、ソフトウェアデバッガ（MSX-SBUG など）を使って、デバッグ作業を行うときは、スタックがこのように設定されていないことがあるので、注意が必要です。

コマンドプロセッサの実体は「COMMAND.COM」というプログラムで、TPAの内側にあります。コマンドプロセッサが必要なのは、プログラムの実行が終わって、次のコマンドを入力するときで、プログラムの実行中は、コマンドプロセッサの機能は必要ないので、プログラムでコマンドプロセッサがロードされている領域を使用（破壊）してもかまわないわけです。このため、ウォームスタートの際には、コマンドプロセッサがプログラムによって壊されている可能性があるため、いったんMSXDOS.SYSが制御を受け取ります。MSXDOS.SYSは、チェックサムを用いてCOMMAND.COMが壊されているかどうかを調べてから、コマンドプロセッサに制御を渡します。COMMAND.COMが破壊されてたときは、再びディスクからロードしますが、破壊されていないときは、ディスクからのロードは行われないので、コマンドレベルに戻るのが早くなります。

こうして、再びコマンドプロセッサが次のコマンド行の入力待ちになります。

3.3 MSX-DOSの入出力

3.3.1 周辺装置との入出力

MSX-DOSの持つ機能は、ディスク上のファイルを扱うものだけではありません。それだけでは、オペレーティングシステムの持つ重要な機能の1つであるユーザーインターフェイスの機能が限られてしまいます。そのため、MSX-DOSでは、コンソールスクリーン、キーボード）やプリンタなどの周辺装置の制御を行う機能を豊富に用意しています。これらの装置は、基本的には1文字単位で入出力を行うものですが、コンソールに関しては、各種のオプションを持った複数の入出力機能に加えて、行入力や行出力の機能も用意しています。

3.3.2 ファイルの入出力

ファイルの入出力（アクセス）の特徴は、データの位置をディスク上のアドレスのような具体的数値で表現するのではなく、「名前」を用いて指定できるという点にあります。MSX-DOSはディスク上に書かれたディレクトリなどの情報を元に、ファイルのデータがどこにあるかということを常に把握しています。目的のデータがディスク上のどんなアドレスに存在しているか、ということはすべてMSX-DOSにまかせ、プログラムはただファイル名を指示するだけで、ファイルのアクセスができます。

1. FCB（ファイルコントロールブロック）

どのファイルをアクセスするか、ということを指示するために、MSX-DOSではFCB(ファイルコントロールブロック）を使います。

FCBは、ファイルを扱う際必要となる情報を格納しておく領域で、1つのファイルを扱うごとに1つ、図3.10や図3.11のようなメモリエリアをプログラムで用意します。

FCBはプログラムで許された範囲でメモリ上のどこに置いても構いませんが、MSX-DOSの機能を活かすため、しばしば5CH番地が使われます。

FCB 先頭からのバイト数 ↓	
0	ドライブ番号 ファイルの存在するディスクドライブを示します(0=デフォルトドライブ、1=A、2=B、~7=H)。
1 ⋮ 8	ファイル名 最大8文字まで指定できます。 8文字に満たない部分はスペース(20H)で埋めます。
9 ⋮ 11	拡張子 最大3文字まで指定できます。 3文字に満たない部分はスペース(20H)で埋めます。
12	カレントブロック シーケンシャルアクセスの際、 参照中のブロック番号を示します(ファンクション14H、15H参照)。
13*	
14	レコードサイズ ブロック中のレコード数が設定されます(ファンクション14H、15H参照)。
15*	
16 ⋮ 19	ファイルサイズ ファイルの大きさがバイト単位で設定されます。
20 21	日付 最後にファイルに書き込みを行った日付が設定されます。
22 23	時刻 最後にファイルに書き込みを行った時刻が設定されます。
24	デバイスID 周辺装置・ディスクファイルの区別が設定されます。
25	ディレクトリロケーション 何番めのディレクトリエントリに該当するファイルであるかが設定されます。
26 27	先頭クラスタ ファイルの先頭のクラスタ番号が設定されます。
28 29	最終クラスタ 最後にアクセスされたクラスタ番号が設定されます。
30 31	相対位置 最後にアクセスされたクラスタの、先頭クラスタからのファイル内での相対位置が番号で設定されます。
32	カレントレコード シーケンシャルアクセスの際、参照中のレコード番号を示します(ファンクション14H、15H参照)。
32*	
33 ⋮ 35	ランダムレコード ランダムアクセスの際、アクセスしたいレコードを指定します。(ファンクション21H、22H、28H参照)。

(*システムで使用)

図3.10 FCBとして使うメモリエリア(CP/M互換)

FCB 先頭からのバイト数 ↓	
0	ドライブ番号 ファイルの存在するディスクドライブを示します(0 =デフォルトドライブ、1 = A、2 = B、〜7 = H)。
1 ⋮ 8	ファイル名 最大 8 文字まで指定できます。 8 文字に満たない部分はスペース (20H) で埋めます。
9 ⋮ 11	拡張子 最大 3 文字まで指定できます。 3 文字に満たない部分はスペース (20H) で埋めます。
12 ⋮ 13*	
14 15	レコードサイズ レコードサイズがバイト数で設定されます (ファンクション 26H、27H 参照)。
16 ⋮ 19	ファイルサイズ ファイルの大きさをバイト単位で設定します。
20 21	日付 最後にファイルに書き込みを行った日付が設定されます。
22 23	時刻 最後にファイルに書き込みを行った時刻が設定されます。
24	デバイス ID 周辺装置・ディスクファイルの区別が設定されます。
25	ディレクトリロケーション 何番めのディレクトリエントリに該当するファイルであるかが設定されます。
26 27	先頭クラスタ ファイルの先頭のクラスタ番号が設定されます。
28 29	最終クラスタ 最後にアクセスされたクラスタ番号が設定されます。
30 ⋮ 31	相対位置 最後にアクセスされたクラスタの、先頭クラスタからのファイル内での相対位置が番号で設定されます。
32*	
33 ⋮ 36	ランダムレコード ランダムブロックアクセスの際、アクセスしたいレコードを指定します。レコードサイズが 1〜63 のときは+33〜36 の 4 バイトを使い、レコードサイズが 64 以上のときは+33〜35 の 3 バイトを使用します (ファンクション 26H、27H 参照)。

(*システムで使用)

図 3.11 FCB として使うメモリエリア (MSX-DOS 専用)

2. ファイルのオープン

FCB を用いてファイルの入出力を行うには、最初にファイルを「オープン」する手続きが必要です。「オープン」とは、ファイル名フィールドだけが設定された不完全な FCB を、ディスク上に記された情報を用いて、完全な FCB に変換することを意味します。図 3.12 は「オープンされていない FCB」と「オープンされた FCB」の違いを表しています。

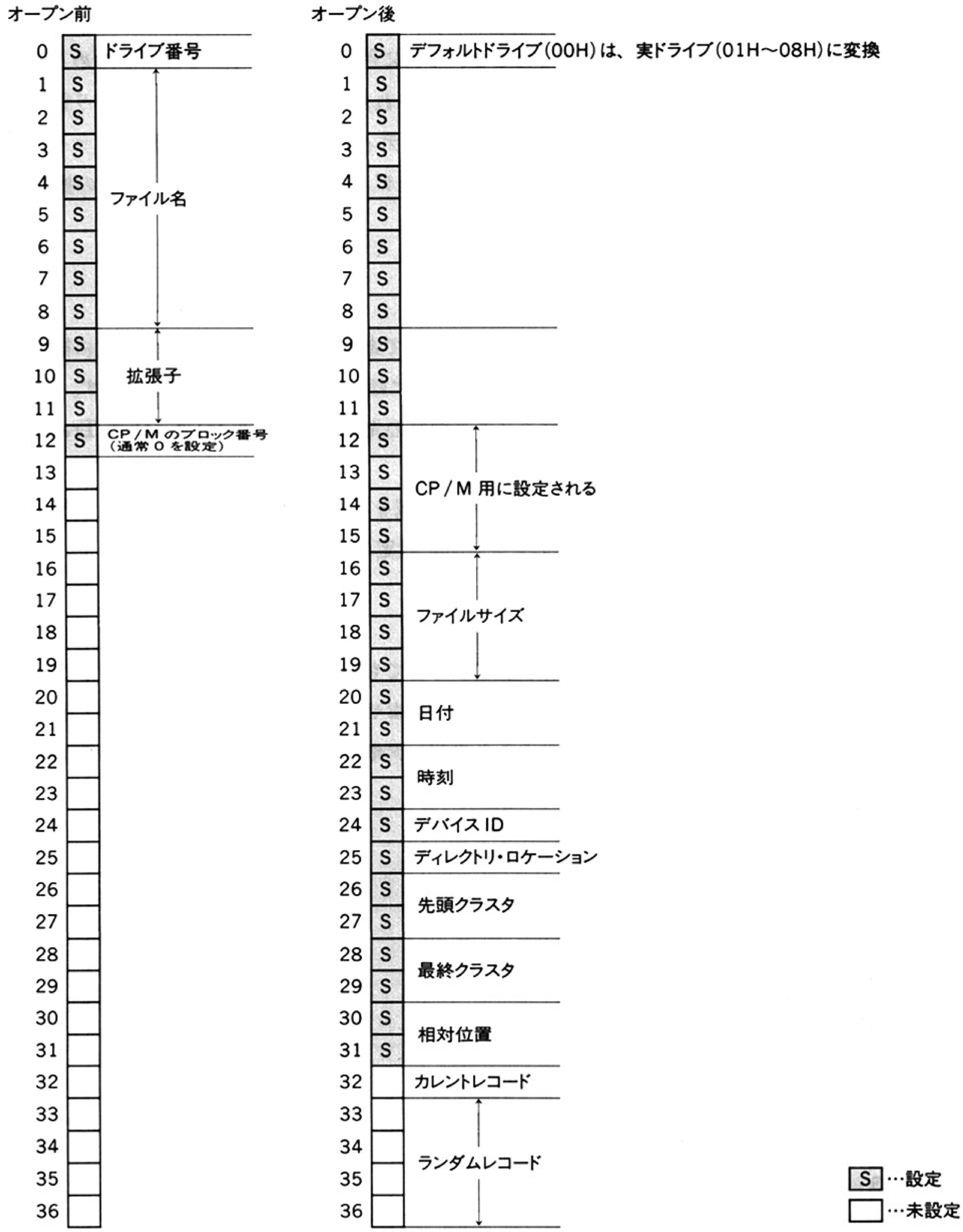


図 3.12　FCB オープンの前後

3. ファイルのクローズ

ファイルをオープンして書き込みを行ったとき、それに伴って、ファイルサイズを始めとするFCBの各フィールドの内容が変更されます。この更新されたFCBの情報をディレクトリ領域に戻しておかないと、次回ファイルをアクセスする際に、ディレクトリの情報と実際のファイルの内容が異なってしまいます。更新されたFCBの情報をディレクトリに戻すというこの操作が「ファイルのクローズ」です。

4. DTA（ディスク転送アドレス）

ディスクの入出力では、基本的には文字単位の入出力である周辺装置と違って、一度に複数のバイトを読み書きします。そのため、ディスクの入出力には「バッファ」が必要になります。MSX-DOSでは、このバッファとして使用されるメモリ領域の先頭アドレスをDTA（Disk Transfer Address＝ディスク転送先アドレス）と呼びます。

ディスクの入出力では、DTAを先頭とするメモリ領域をバッファとして使用します。DTAの値は、初期設定では0080Hですが、任意のアドレスに設定し直すことができます。

5. レコード

ファイルの大きさは、ディスク容量の許す限りいくらでも（最大1Gバイト）大きくすることができるので、メモリに全体が読み込めることは期待できません。そのため、ファイルの入出力では、ファイルをレコードと呼ばれる適当な大きさのデータ単位に分割して、レコード単位で読み書きする方法がとられています。

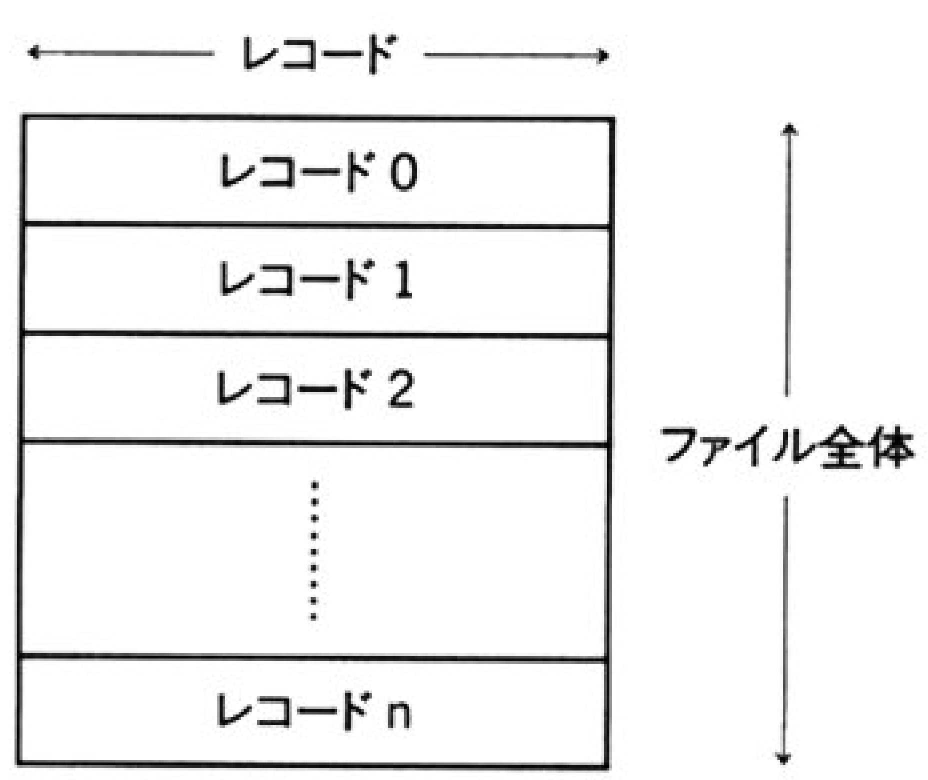


図3.13 ファイルとレコード

したがって、ファイルのアクセスを行うためには、さらにどのレコードをアクセスするかという指示が必要になります。このレコードの指示方法はファンクションコールの種類により、大きく2つのグループに分けられます。1つのグループはCP/Mとの互換性を保つために用意された入出力で、もう1つのグループはMSX-DOSの独自のランダムブロック入出力です。この2つ

は、FCBの使用方法が異なっているので、同じFCBに対して同時に使用することはできません。

6. CP/M互換のレコードアクセス

CP/Mとの互換性を保つため、MSX-DOSではCP/Mと同様のファイルアクセス方式をサポートしています。CP/Mでは、2つの全く異なるアクセス方式があります。また、レコードサイズは128バイトに固定されています。

■シーケンシャルアクセス（カレントレコード＋カレントブロックによるレコード管理）　CP/Mとの互換性を保つための1つめの方法が、「カレントレコード」と「カレントブロック」で管理されるシーケンシャルアクセスです。ファイルのアクセスは常に先頭から順（シーケンシャル）に行い、アクセスしたレコード数はFCBのカレントレコードフィールドでカウントされます。カレントレコードフィールドの値は128になると0にリセットされ、その桁上がりがカレントブロックフィールドにカウントされます。このアクセス方式では、「ランダムレコード」は使いません。

■ランダムアクセス（ランダムレコードによるファイル管理）

CP/Mとの互換性を保つための2つめの方法が、「ランダムレコード（3バイト）」で管理されるランダムアクセスです。「ランダムレコード」を変更することで、任意の位置のレコードをアクセスすることができますが、別のレコードをアクセスするためには、毎回この操作が必要です。

7. ランダムブロックアクセス（論理レコードによるファイル管理）

MSX-DOSには、「ランダムブロックアクセス」という、大変に有能な入出力方式があります。これは、MSX-DOS独自のファンクションコールで、次のような特徴があります。

- ■レコードサイズを任意に設定できる。
- ■複数のレコードを一度にアクセスできる。
- ■シーケンシャルアクセスとランダムアクセスを任意に切り換えられる。

ランダムブロックアクセスでは、FCBの「レコードサイズ」を用いて、レコードの大きさを指示し、レコード位置を「ランダムレコード」で指示します。レコードサイズの2バイトには任意の値を入れることができます（1〜65535）。ファイル全体を1レコードとして扱うことも、1バイトを1レコードとして扱うことも、128バイトを1レコードとして扱うこと（CP/M方式）も可能です。

このアクセス方式では、ランダムレコードのフィールドがアクセスの終わったレコードの次のレコードを指すように、常に更新されます。したがって、ランダムレコードを1度セットするだけで、任意のレコードからレコードまでをシーケンシャルに読み書きすることが簡単にできます。

3.3.3 日付と時刻の設定

MSX-DOSは、ファイルの内容が変更されたときには、自動的にファイルの最終更新日付・時刻を現在の日付・時刻に書き換えます。これは、MSX-DOS自身には何の意味もないのですが、ユーザーがファイルを管理する目的には、非常に有用です。このため、MSX-DOSには、日付・時刻の読み出しや変更を行うファンクションコールがあります。

3.4 ディスクファイルの構造

ファンクションコールを使用すれば、手軽にキメ細かくファイルを取り扱うことができます。ユーザーは、ディスク上でファイルがどのような方法で管理され、どのような形式で記録されているか、といった情報を詳しく知る必要はありません。しかし、場合によっては、ディスクの管理情報を取り出したり、それをもとにファイルとは無関係にディスクを直接アクセスしたりする手段が必要になることがあります。

MSX-DOSでは、このような目的のために、ディスクの管理情報を得たり、「論理セクタ」を直接アクセスするファンクションコールを用意しています。

3.4.1 ディスク上のデータ構造

セクタを直接アクセスする場合には、どのセクタにどういった情報が書き込まれているか、という基本知識が必要になります。

1. 論理セクタ

MSX-DOSでは、3.5インチフロッピーディスクでもハードディスクでも、あるはその他のドライブでも基本的にはアクセスすることができます。それぞれのドライブやメディアの種類によりセクタの大きさ、トラック毎のセクタ数、記録面の数などは違っていますが、これを統一的に管理するため、MSX-DOSでは、ディスク上の物理的な境界にとらわれず、すべてのセクタに連続した通し番号をつけて、その番号でセクタを管理する方法を採用しています。これを「論理セクタ」と呼びます。

「論理セクタ」(以下、セクタ)の番号は0からそのディスクの総セクタ数-1(ディスクの種類によって異なる)までの一連の番号によって指定します。

MSX-DOSでは、ディスクの中のセクタを表3.5に示す4つの領域に分けています。最初の3つの領域にはデータを管理するための情報が書き込まれ、ファイルデータの本体は「データ領域」の部分に書き込まれます。これらの位置関係は図3.14のとおりです。ブートセクタはかならずセクタ0にありますが、

- FAT
- ディレクトリ
- データ領域の開始セクタの位置

はメディアによって異なります。ただし、これらの情報は、ブートセクタ読み出せば得ることができます。

表3.5 ディスクの領域

領 域	内 容
ブートセクタ	ディスク固有の情報とMSX-DOSの起動プログラム
FAT	ディスク上のファイルの位置情報
ディレクトリ	ディスク上のファイルの管理情報
データ領域	実際のファイルデータ

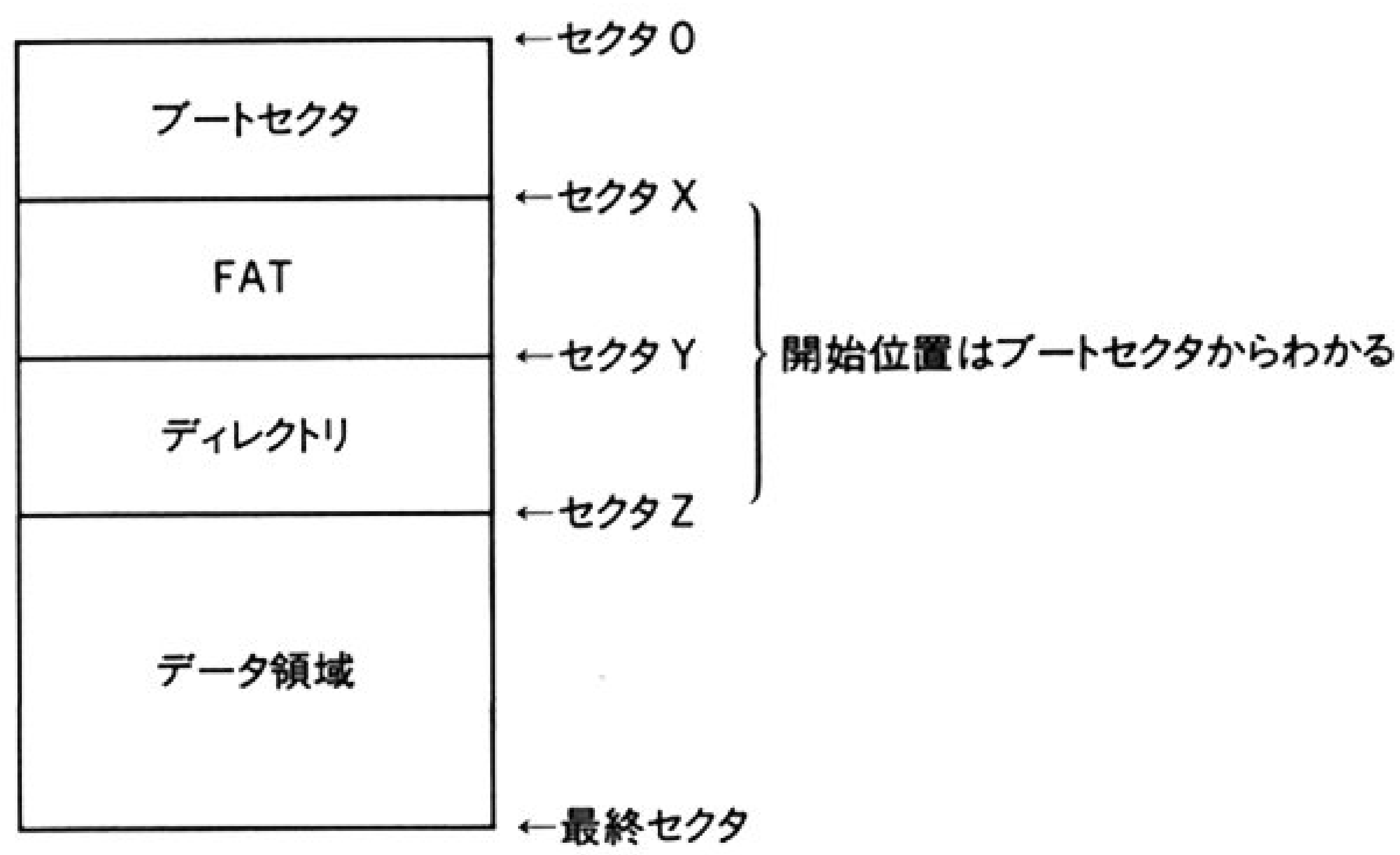


図3.14 ディスク上の領域の位置関係

2. クラスタ

ディスクの入出力は、前述のとおりセクタが基本単位です。ただし、ファイルに対してディスク上のセクタを割り当てるときには、セクタではなく複数のセクタから成る「クラスタ」という単位が使われます。それぞれのファイルには、そのファイルサイズに応じて必要な数のクラスタ

が割り当てられます。1クラスタよりも大きなサイズのファイルの場合、データは複数のクラスタにまたがって記録されます。1クラスタ未満の部分については、たとえそれが1バイトであっても1クラスタ分のデータ領域が割り当てられます。クラスタは論理セクタと同様に連続した番号で指定されていますが、FATの項で述べる理由で、2から始まる通し番号になっており、データ領域の先頭がクラスタ#2の位置に相当します。

3. ブートセクタとDPB(ドライブパラメータブロック)

MSX-DOSでは、接続されている個々のドライブごとに「DPB」という領域がメモリ上のワークエリアに設けられ、各ドライブに固有の情報が記録されます。MSX-DOSはどのようなタイプのディスクドライブにも対応ができますが、それはこのDPBを参照して個々のドライブに対応した処理を行うことによって、メディア間の差異が吸収できるからです。

DPBに書き込まれる情報は、ブートセクタに記録されているもので、それがMSX-DOSの起動時やメディアが交換される毎に変更されます。ただし、ブートセクタとDPBは、図3.15と図3.16に示すように、形式が異なっています。

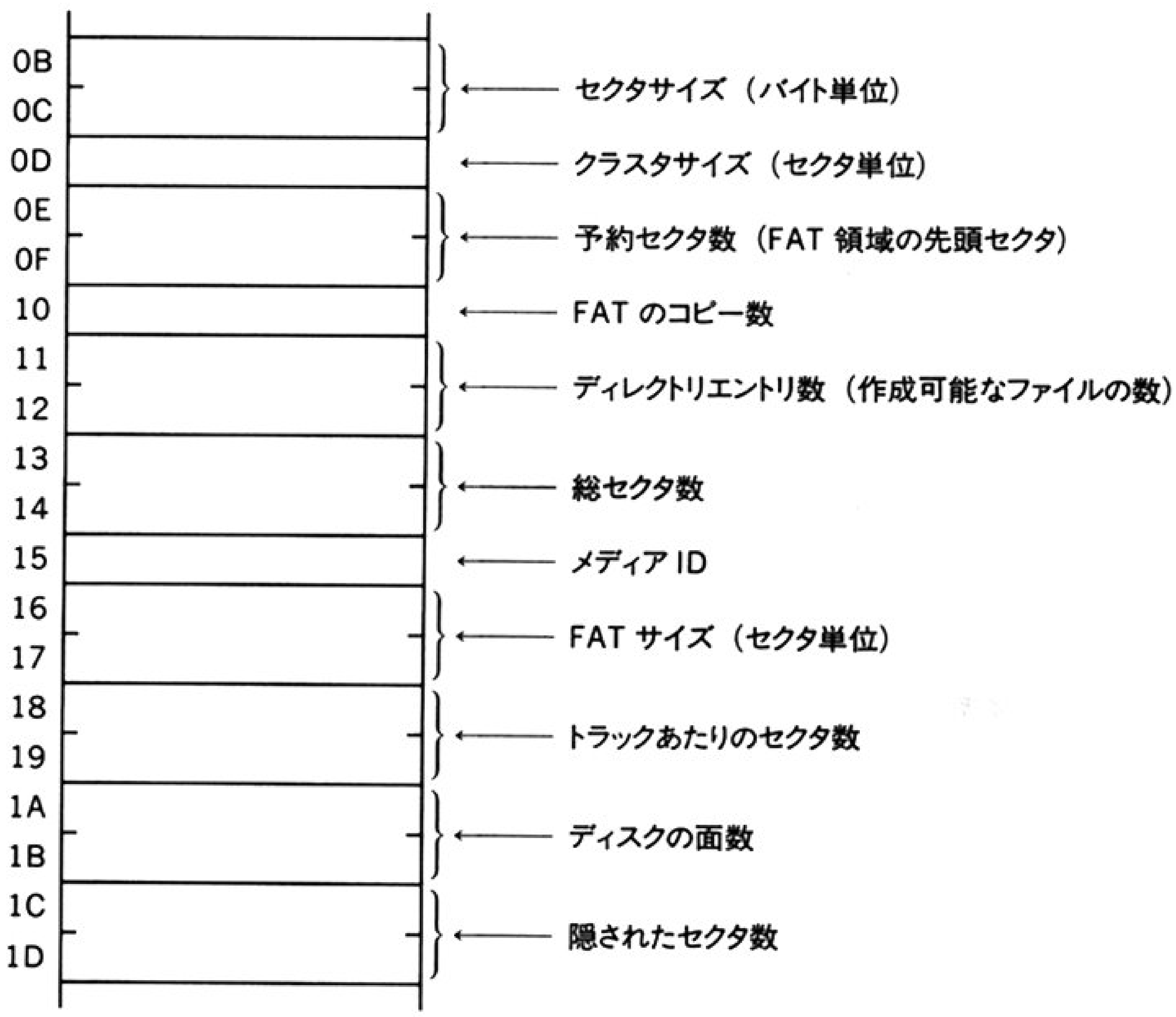


図3.15 ブートセクタの情報

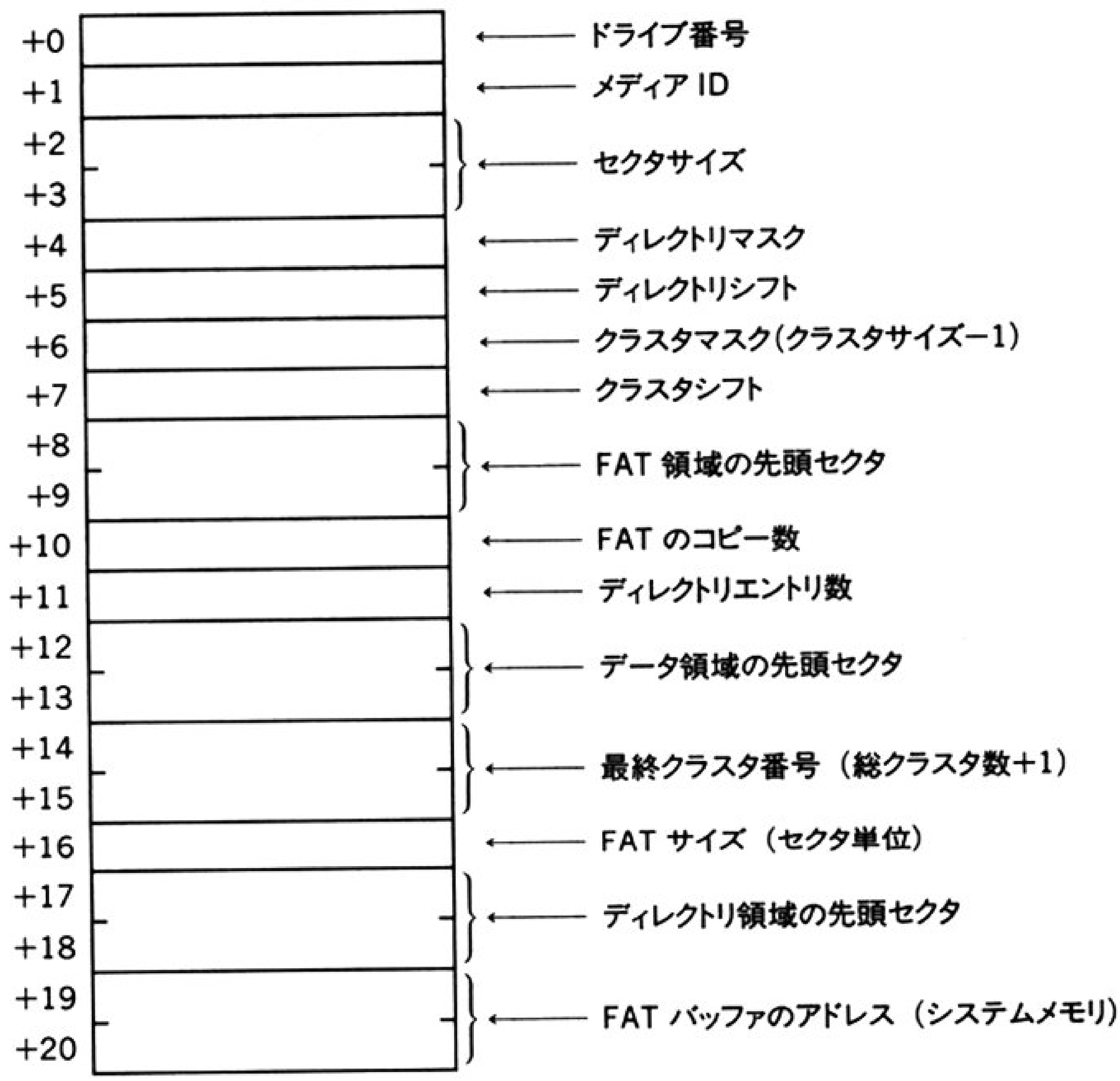


図 3.16　DPB の構造

4. FAT（ファイルアロケーションテーブル）

MSX-DOS では、1クラスタよりも大きなサイズのファイルは、複数のクラスタにまたがって記録されますが、そのとき連続した番号のクラスタが使用されるとは限りません。特に、ファイルの作成・削除を何度も繰り返した後では、使われなくなったクラスタがディスク上のあちこちに散在した状態になります。この状態でサイズの大きなファイルを作成すると、データは飛び飛びのクラスタに分散して置かれます。そこで、「何番目のクラスタは何番目のクラスタに続いている」、というリンク情報を記録しておく場所が必要になります。それが FAT の役割です。また、未使用クラスタの位置や、不良クラスタが発見されたときに以後そこをアクセスしないように記録する目的にも FAT が利用されます。

FAT に記録されるこのようなクラスタのリンク情報や不良クラスタ情報は、ディスクファイルを管理するうえで不可欠なものであり、一部でも破損してしまうとディスク全体が使用できなくなる恐れがあります。そのため FAT は常に複数個のコピーが用意され、万一に備えています。

FAT の例を図 3.17 に示します。FAT には、

- 先頭の1バイトは「FAT ID」と呼ばれ、ディスクのメディアタイプを示す値
- 次の2バイトはダミー値の FFH
- その次から、1クラスタにつき 12 ビットというフォーマットで実際のリンク情報を記録

していた FAT エントリ

が記録されます。0番と1番に相当する3バイトが「FAT ID」に使われていますので、ファイルのデータに対応するFATエントリは2番から始まります。FATエントリの番号は、それに対応するクラスタの番号でもあります。FATエントリに記録された12ビットのリンク情報は、図3.18のように並んでいます。リンク情報は、次に続くクラスタ番号を示す値です。もし、FFFHとなっているときは、そのクラスタでファイルが終了したことを意味します。

図3.17の例では、クラスタ#2→クラスタ#3→クラスタ#4という3クラスタ分の大きさのファイルと、クラスタ#5→クラスタ#6の2クラスタ分のファイルが存在していることがわかります。なお、クラスタが番号の小さい順にリンクしているのは図を見やすくするためで、実際には番号順であるとは限りません。

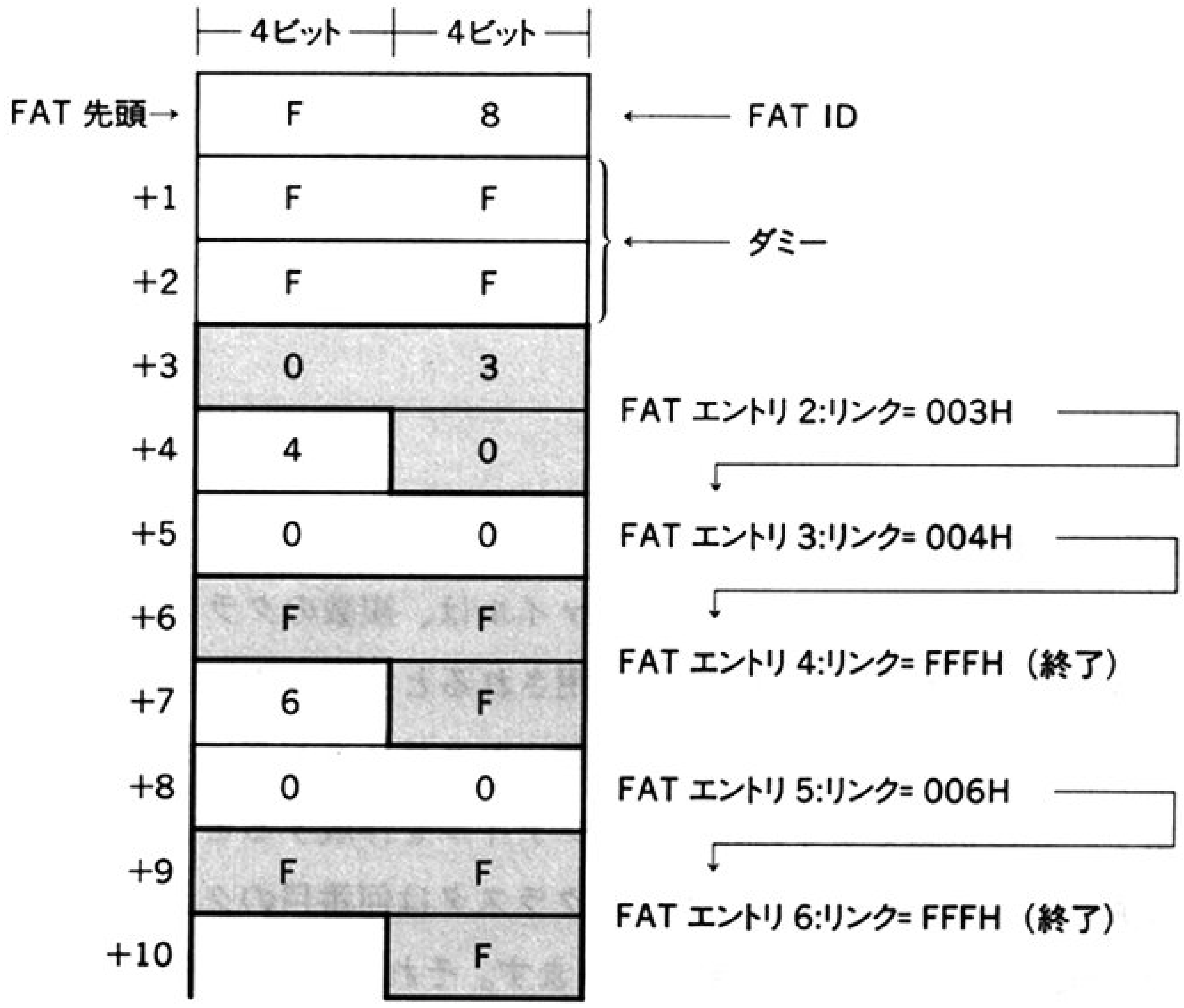


図3.17 FATの実例

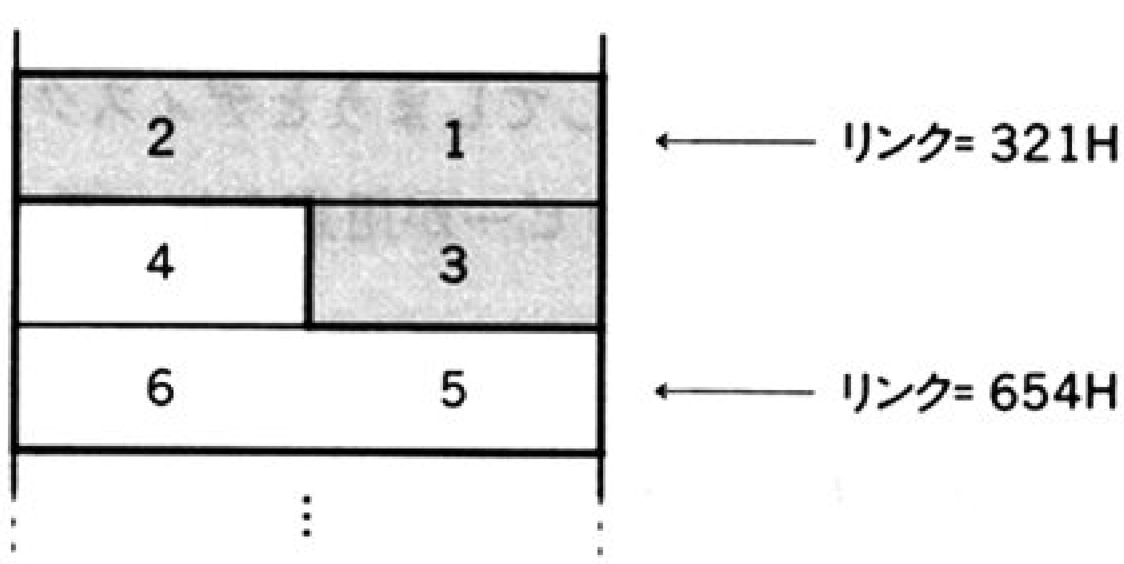


図3.18 FATの読み方

5. ディレクトリ

FAT はデータの位置関係などを表すものであり、ファイル自体に関する情報は含んでいません。したがって、そのファイルの名前やそれに付随する情報を知るには、FAT とは別の情報源が必要です。これが「ディレクトリ」です。

ディレクトリはディスク上のディレクトリ領域に記録されていて、図3.19のように32バイトごとにディレクトリエントリ（ディレクトリの格納場所）が並んでいます。ファイルを作成すると、使われていないディレクトリエントリの中で、いちばん番号が小さいところに目的のファイルのディレクトリが作られます。ファイルが削除されると、該当するディレクトリエントリの最初の1バイトにE5H が書き込まれ、そのディレクトリエントリが空いたことを示します。ディレクトリエントリがすべて使用されてしまうと、データ領域がいくら残っていても新しいファイルを作ることはできません。

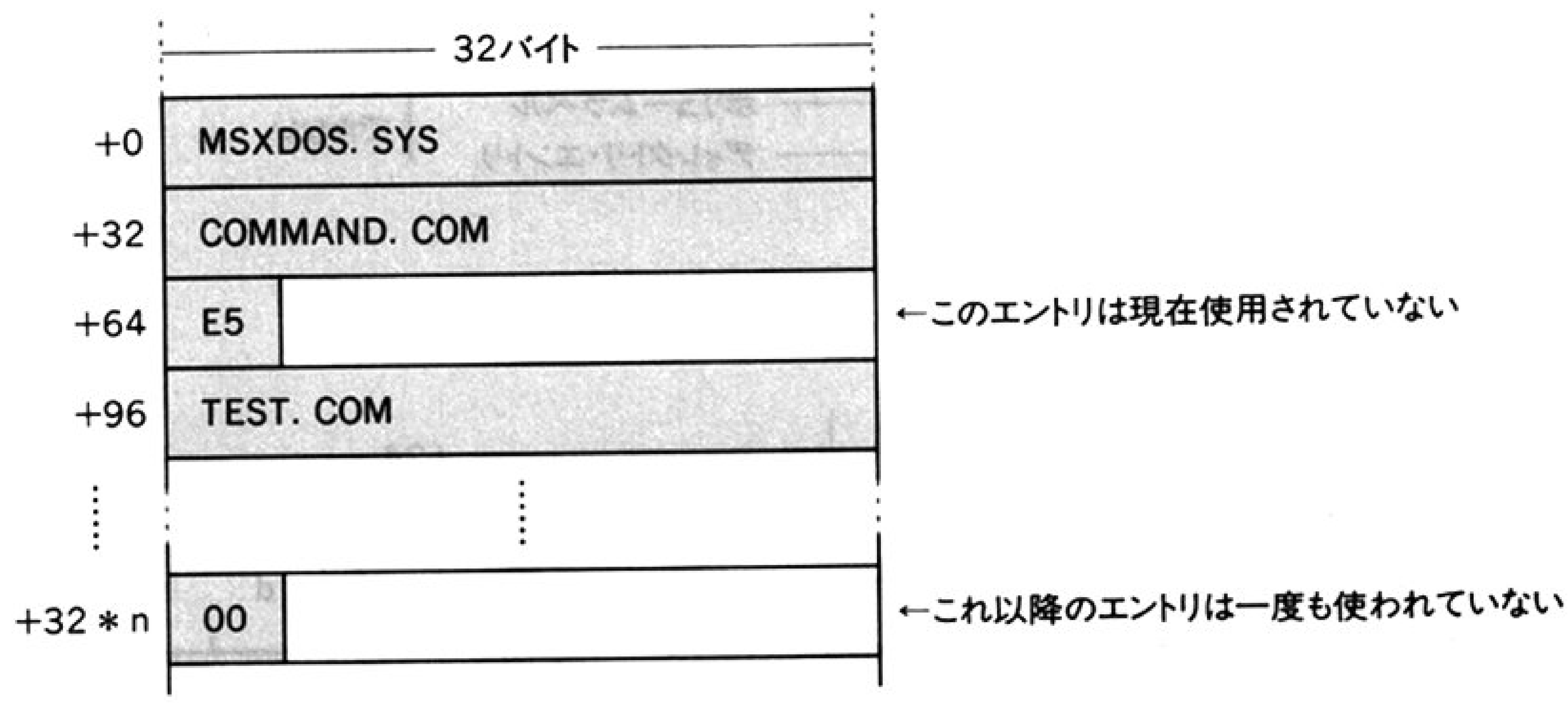


図3.19　ディレクトリ領域の構造

ディレクトリエントリは図3.20のような構造で、それぞれファイル名、ファイル属性、作成・更新の日時、ファイルの先頭クラスタ番号、ファイルサイズの情報を記録しています。

ファイル属性は、ファイルに各種の属性を与えるものです（図3.21参照）。MSX-DOS では属性を持ったファイルを作ることはできません。ただし、既存のファイルに不可視・システム・ボリューム・ディレクトリのいずれかの属性が付けられていると、そのファイルはファンクションコールではアクセスできないようになります。読み出し専用（書き込み禁止）属性は無視されます。

日付と時刻は図3.22と図3.23のように、それぞれ2バイトの領域を3つのビットフィールドに分割して記録しています。「年」は7ビットに0～99の値を設定することで、西暦1980年～2079年を表します。「秒」用のビットフィールドは5ビットで、時間の分解能は2秒です。

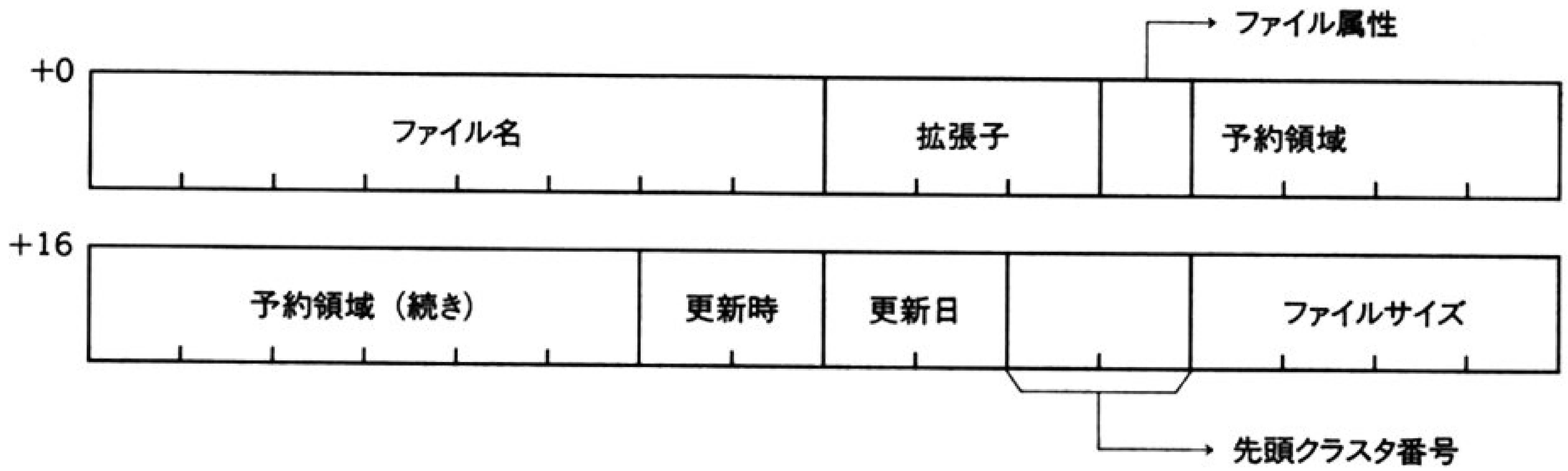


図3.20　ディレクトリの構造

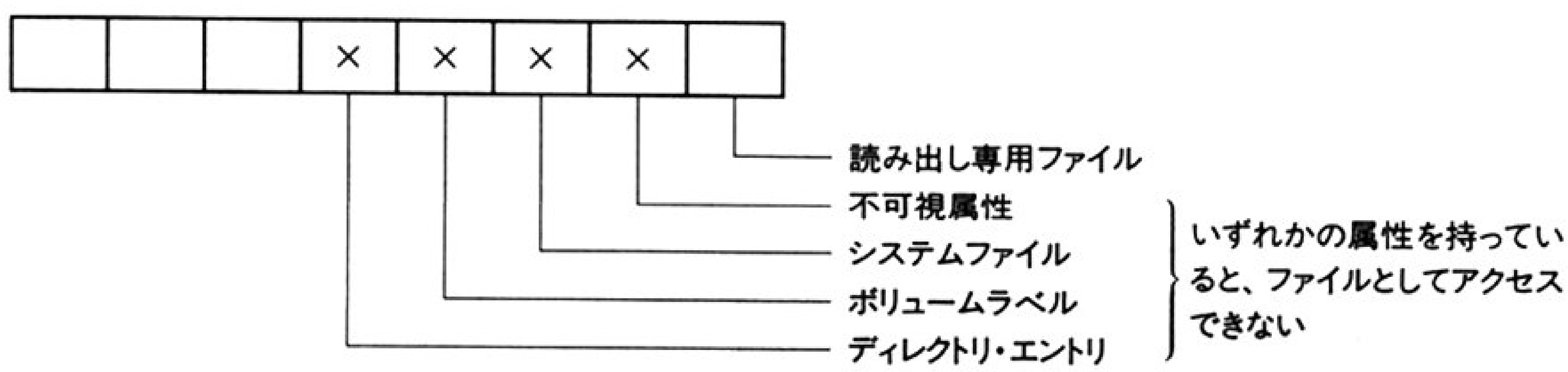


図3.21　ファイル属性

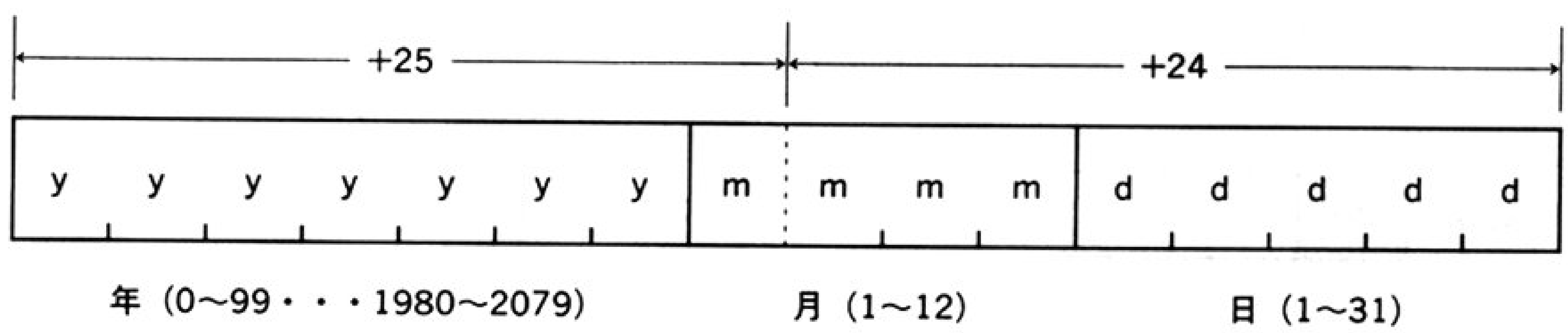


図3.22　日付を表すビットフィールド

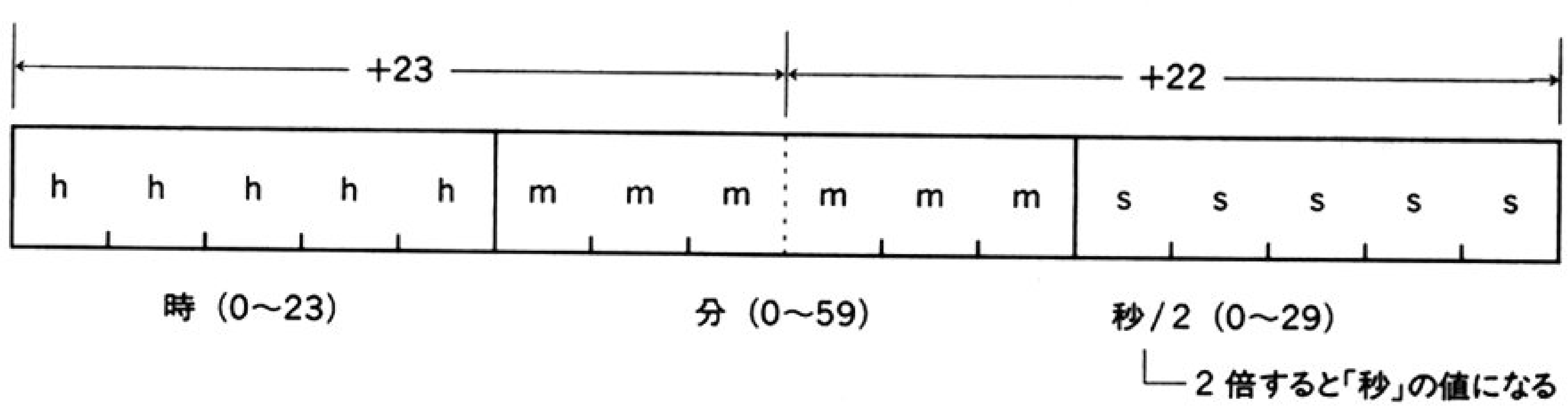


図3.23　時刻を表すビットフィールド

6. クラスタからセクタへの換算

FATやディレクトリでは、ディスク上のデータの位置はクラスタ単位で表わされています。クラスタで示されたこれらのデータをファンクションコールでアクセスするためには、あるクラスタが何番のセクタに対応しているか、という関係を求めなければなりません。これは、データ領域がクラスタ#2から開始していることを元に、以下のように計算することができます。

1. 与えられたクラスタ番号をCとする。
2. データ領域の開始セクタを調べ、これをS0とする。
3. 1クラスタが何セクタに相当するか調べ、これをnとする。
4. 求めるセクタ番号Sは、S＝S0＋(C-2)＊nの計算で得られる。

4章 ファンクションコール

4.1 概要

MSX ディスクシステムの持つ入出力機能は BDOS というかたちで、汎用のサブルーチン形式で ROM にまとめられています。ファンクションコールとは、BDOS を呼び出すためにあらかじめ決められた手順のことで、ユーザープログラムはファンクションコールを実行することでファイルの入出力操作が簡単にできます。

ファンクションコールの役割は、大きく分けると 2 つがあります。

1. 基本的な機能をあらかじめ用意することによってプログラムの負担を少なくすること。
2. すべてのプログラムが共通の手順を使用することによって、移植性や汎用性を高めること。

ファンクションコールを自由自在に活用することによって、プログラム開発の期間は短縮され、できあがったプログラムは移植性の高いものになります。これは、OS を利用した時の最大のメリットの 1 つです。

ファンクションコールを実行するには、CPU の C レジスタに決められたファンクション番号を入れ、次のアドレスをコールします。

0005H	MSX-DOS 上の外部コマンド
F37DH	Disk BASIC 上のマシン語プログラム

例えば、ファンクション番号が 1FH であり、コール手順として A レジスタに 00H をセットすることになっているファンクションコールがあるとすると、このファンクションコールは次のようにして実行します。

```
ld      a, 00h
ld      c, 1fh
call    0005h           ; Disk BASIC のときは 0F37DH
```

CALL 文の後には、戻り値（処理の結果）を受け取ったり、退避していたレジスタを復帰させたりする事後処理が続きます。

Disk BASIC のプログラムからは、CLEAR 文で確保した領域にマシン語プログラムを置き、その開始アドレスを USR 関数で呼び出すことでファンクションコールを利用できます。

4.2 書式

ここでは、以下の書式によってファンクションコールの使用法を紹介します。

ファンクション

ファンクション番号とは、ファンクションコールの種類を判別するためのもので、それぞれのファンクションコールに対応する番号が決められています。ファンクションコールを実行するときには、このファンクション番号を C レジスタにセットします。

コール手順

ファンクションコールの前準備として、レジスタやメモリに必要な値をセットしなければならないことがあります。ここでは、これを「コール手順」の部分に示します。

戻り値

ファンクションコールの結果得られた値は、レジスタやメモリにセットされます。獲得した値が、どこにどのようにセットされるかを、「戻り値」の部分に示します。

ただし、ファンクションコールでは、戻り値がセットされるレジスタ以外にも、ファンクション内での作業に使用され、内容が破壊されるレジスタがあります。ファンクションコールでは原則としてレジスタのもとの内容は保証していません。そこで、ファンクションコールを使用するときには、その前に、破壊されては困るレジスタの内容を、適当な場所（スタックなど）に退避して下さい。

解　説

ファンクションコールの解説です。

表3.6 ファンクションコール一覧

ファンクション番号	機　能	CP/M*	ページ
00	システムリセット	○	419
01	コンソール1文字入力	○	419
02	コンソール1文字出力	○	420
03	補助入力装置1文字入力	○	421
04	補助出力装置1文字出力	○	421
05	プリンタ1文字出力	○	422
06	コンソール直接入出力（入力時エコーバック、コントロールコードチェック、入力待無）	○	422
07	コンソール直接入力（エコーバック、コントロールコードチェック無）		423
08	コンソール1文字入力（エコーバック無）		423
09	コンソール文字列出力	○	424
0A	コンソール1行入力	○	424
0B	コンソール入力チェック	○	425
0C	バージョン番号の取り出し	○	425
0D	ディスクリセット	○	426
0E	ディフォルトドライブの設定	○	426
0F	ファイルのオープン	○	427
10	ファイルのクローズ	○	427
11	ファイルの検索（ワイルドカードに一致する最初のファイル）	○	428
12	ファイルの検索（ワイルドカードに一致する後続のファイル）	○	429
13	ファイルの削除	○	429
14	シーケンシャル読み出し	○	430
15	シーケンシャル書き込み	○	430
16	ファイルの作成	○	431
17	ファイル名の変更	○	432
18	ログインベクトルの獲得	○	432
19	ディフォルトドライブの獲得	○	433
1A	転送アドレスの設定	○	433
1B	ディスク情報の獲得		434
21	ランダム読み出し	○	434
22	ランダム書き込み	○	435
23	ファイルサイズの獲得	○	435
24	ランダムレコードの設定	○	436
26	ランダムブロック書き込み		436
27	ランダムブロック読み出し		437
28	ゼロ書き込みを伴うランダム書き込み	○	438
2A	日付の獲得		438
2B	日付の設定		439
2C	時刻の獲得		440
2D	時刻の設定		440
2E	ベリファイフラグの設定		441
2F	論理セクタの読み出し		441
30	論理セクタの書き込み		442

注　意　・「○」は、CP/M 2.2のファンクションとの互換性があるものを示している。

ファンクションコールのファンクション番号は00H～30Hですが、この中には以下の無効なファンクション番号があります。

■ 1CH～20H、25H、29H

これらの無効なファンクションコールを実行したときには、AレジスタとBレジスタに00Hがセットされる以外は何も行いません。また、31H以降のファンクション番号でも結果は同じです。

4.3 ファンクションコールの解説

システムリセット

ファンクション

00H

コール手順

なし

戻り値

なし

解　説

MSX-DOS上のプログラムからコールしたときには、MSX-DOSがウオームスタートし、DOSのコマンドレベルに戻ります。Disk BASIC上のプログラムからコールしたときには、Disk BASICがウォームスタートし、BASICのコマンドレベルに戻ります。このとき、ロードされているBASICのプログラムは破壊されません。

コンソール入力

ファンクション

01H

コール手順

なし

戻り値

A　　コンソールから入力した1文字

解　説

入力がないとき（キーボードバッファが空のとき）には、入力を待ちます。入力された文字はコンソールにエコーバックされます。以下に示すコントロールキャラクタはファンクション内部で処理されるため、入力として扱いません。

[CTRL]+[C]	システムリセット
[CTRL]+[P]	プリンタへのエコー開始
[CTRL]+[N]	プリンタへのエコー停止

[CTRL]+[C]が入力されるとプログラムの実行を中断してMSX-DOSのコマンドレベルに戻ります。[CTRL]+[P]が入力されると、以降のコンソール出力はすべてプリンタにもエコーします。この状態は、[CTRL]+[N]を入力することによって解除されます。

コンソール出力

ファンクション

02H

コール手順

E　　出力する文字コード

戻り値

なし

解　説

Eレジスタで指定した文字を画面に表示します。この際、コンソール入力をチェックし、[CTRL]+[C]、[CTRL]+[P]、[CTRL]+[N]、[CTRL]+[S]の処理を行います。[CTRL]+[S]が入力されると、次に任意のキーを押すまでファンクション内部で入力待ちを行うので、プログラムの一時停止ができます。

補助入力

ファンクション

03H

コール手順

なし

戻り値

A　　補助入力装置（AUX デバイス）から読み込んだ1文字

解　説

補助入力装置がセットされていないときは、EOF 文字 (1AH) を返します。コンソール入力は、ファンクション 02H と同様にチェックします。

補助出力

ファンクション

04H

コール手順

E　　補助出力装置（AUX デバイス）に出力する文字コード

戻り値

なし

解　説

補助出力装置がセットされていないときは何もしません。出力は捨てられます。コンソール入力は、ファンクション 02H と同様にチェックします。

プリンタ出力

ファンクション

05H

コール手順

E　　プリンタに出力する文字コード

戻り値

なし

解　説

文字コードをプリンタに出力します。コンソール入力は、ファンクション 02H と同様にチェックします。

直接コンソール入出力

ファンクション

06H

コール手順

E	FFH	入力
	FFH 以外	セットされた値を文字コードとみなしてコンソールに出力

戻り値

■ E レジスタが FFH だったとき（入力）

A　　コンソール入力があるときはその文字コード
　　　コンソール入力がないときは 00H

■ E レジスタが FFH 以外だったとき（出力）

なし

解　説

入力のとき、エコーバック、コントロールキャラクタの処理は行いません（入力文字として扱われる）。また、出力のとき、プリンタへのエコーバックは行いません。

直接コンソール入力

ファンクション

07H

コール手順

なし

戻り値

A　　　コンソールから入力した1文字

解　説

エコーバック、コントロールキャラクタの処理は行いません。

このファンクションコールはCP/Mとの互換性はありません。

コンソール入力

ファンクション

08H

コール手順

なし

戻り値

A　　　コンソールから入力した1文字

解　説

エコーバックは行いません。コントロールキャラクタはファンクション01Hと同様に処理します。

このファンクションコールはCP/Mとの互換性はありません。

文字列出力

ファンクション

09H

コール手順

DE　　コンソールに出力する文字列があるメモリの先頭アドレス

戻り値

なし

解　説

文字列の最後には、終端文字（ターミネータ）として「$」(24H) を置きます。最初に現れる「$」の直前までの文字列をコンソールに出力するので、「$」文字を出力することはできません。

コントロールキャラクタはファンクション 02H と同様にチェックします。

文字列入力

ファンクション

0AH

コール手順

DE　　行バッファの先頭アドレス

行バッファ＋0　　最大入力文字数（1〜255）（図 3.22 参照）

戻り値

行バッファ＋1　　実際に入力された文字数

行バッファ＋2〜　　コンソールから入力された文字列

解　説

リターンキーの直前までをコンソールからの入力と見なします。最大入力文字数以上を入力することはできません。このファンクションコールによる文字列入力時には、テンプレートによる編集ができます。

このファンクションコールは、コントロールキャラクタをチェックします。

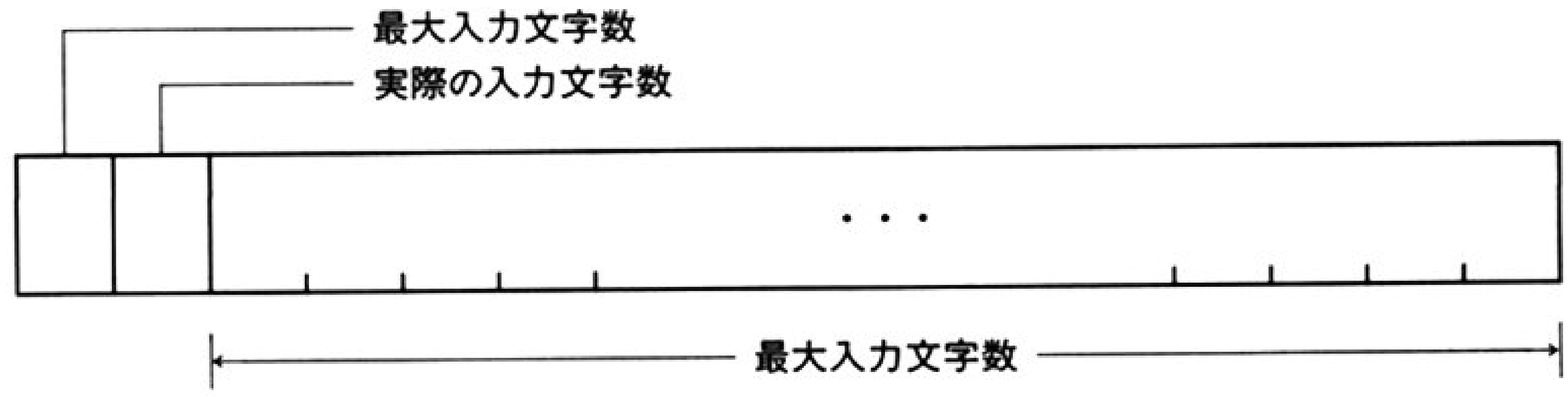


図3.24　行バッファの構造

コンソール入力状態のチェック

ファンクション

0BH

コール手順

なし

戻り値

A	FFH	コンソール入力がある
	00H	コンソール入力がない

解　説

コンソール入力の状態を調べます。入力文字があるときもその文字は取り出しません。ただし、コントロールキャラクタはファンクション02Hと同様にチェックします。入力文字は、ファンクション01Hか08Hで取り出すことができます。

バージョン番号の獲得

ファンクション

0CH

コール手順

なし

戻り値

HL　　0022H

解　説

このファンクションコールは、CP/M のバージョン番号を獲得するためのものです。MSX-DOS では、必ず 0022H が戻ります。

ディスクリセット

ファンクション

0DH

コール手順

なし

戻り値

なし

解　説

変更されて、まだディスクに書き込まれていないセクタがあれば、それをディスクに書き込んだ後、ディフォルトドライブを A ドライブにセットし、DTA を 0080H にセットします。

デフォルトドライブの設定

ファンクション

0EH

コール手順

E　　デフォルトドライブ番号（A = 0、B = 1、・・・ H = 7）

戻り値

なし

解　説

デフォルトドライブを変更します。FCBなどのドライブ番号に「0」が指定されたときに有効です。接続されていないドライブが指定されたときには、変更しません。

ファイルのオープン

ファンクション

0FH

コール手順

DE	オープンされていないFCBの先頭アドレス
FCBのカレントブロック	読み出すブロック番号

戻り値

■オープン成功のとき

A　　0

FCBの各フィールドが設定される

■オープン失敗のとき

A　　0FFH

解　説

FCBがファイル入出力のファンクションコールで使用できるようになります。レコードサイズ、カレントレコード、ランダムレコードの各フィールドは使用するファンクションに応じてオープンの後、ユーザープログラムでが設定しなければなりません。

ファイルのクローズ

ファンクション

10H

コール手順

DE　　オープンされたFCBの先頭アドレス

戻り値

A	00H	クローズが成功
	FFH	クローズが失敗

解　説

現在のFCBの内容をディスク上の該当するディレクトリエリアに書き込むことによって、ファイルの更新に関する整合性を保ちます。ただし、ファイルの内容を変更していないときには、書き込みは行いません。クローズしたFCBはオープンされていないFCBとして再利用できます。また、オープンされたFCBとしてさらに入出力を続けることもできます。

ファイルの検索（最初の一致）

ファンクション

11H

コール手順

DE　　オープンされていないFCBの先頭アドレス

戻り値

A	00H	ファイルが見つかった
	FFH	ファイルが見つからなかった
		ファイルが見つかったときは、DTAで指示される領域にドライブ番号を、それに続く32バイトにそのファイルのディスク上のディレクトリエントリをセットする。

解　説

ファイル名にはワイルドカード文字の「?」を使用することができます。例えば、ファイル名の名前部分に「????????」、拡張子部分に「MAC」を設定すると、「MAC」という拡張子をもつファイルのうち、最初にマッチしたファイルのディレクトリ情報を得ることができます。ファンクションコールレベルでは、ワイルドカード文字の「*」はサポートしていないので、適当な数の「?」に置き換えて下さい。一致するファイルすべてを検索したいときや、一致するファイルが1個だけかどうかを知りたいときには、ファンクション12Hを利用します。

DTA領域に設定されたデータはオープンされていないFCBとして使用できます。

ファイルの検索（後続の一致）

ファンクション

12H

コール手順

なし

戻り値

A	00H	ファイルが見つかった
	FFH	ファイルが見つからなかったとき
		ファイルが見つかったときは、DTA で指示される領域にドライブ番号を、それに続く 32 バイトにそのファイルのディスク上のディレクトリエントリをセットする。

解　説

このファンクションコールは、ファンクション 11H のワイルドカード文字によるファイル名の指定と一致した複数のファイルを検索するときに使います。単独で使用することはできません。このファンクションコールを繰り返し使用して、ファンクション 11H によって一致した複数のファイルのディレクトリ情報を、1 つずつ順番に獲得していくことができます。

DTA 領域に設定されたデータはオープンされていない FCB として使用できます。

ファイルの削除

ファンクション

13H

コール手順

DE　　オープンされていない FCB の先頭アドレス

戻り値

A	00H	削除が成功
	FFH	削除が失敗

解　説

このファンクションコールでは、ファイル名にワイルドカード文字を指定して、複数のファイルを一度に削除することができます。

シーケンシャルな読み出し

ファンクション

14H

コール手順

DE	オープンされた FCB の先頭アドレス
FCB のカレントレコード	読み出すレコード番号

戻り値

A	00H	読み出しが成功
	01H	読み出しが失敗

読み出しが成功したときは、DTA 以降の 128 バイトに読み込んだレコードをセットします。

解　説

FCB のカレントブロックとカレントレコードは、読み出し後に次のレコードを指すように自動的に更新されます。したがって、連続して読み出すときは、カレントブロックとカレントレコードを再設定する必要はありません。

このファンクションコールが失敗するのは、ファイルの終りに達したときです。また、読み出すレコードサイズは 128 バイト固定です。

シーケンシャルな書き込み

ファンクション

15H

コール手順

DE	オープンされた FCB の先頭アドレス
FCB のカレントレコード	書き込むレコード番号
DTA 以降の 128 バイト	書き込むデータ

戻り値

A	00H	書き込みが成功
	01H	書き込みが失敗

解　説

FCB のカレントブロックとカレントレコードは、書き込み後に次のレコードを指すように自動的に更新されます。したがって、連続して書き込むときには、カレントブロックとカレントレコードを再設定する必要はありません。

このファンクションコールが失敗するのは、ディスクの残り容量がなくなったときです。

ファイルの作成

ファンクション

16H

コール手順

DE	オープンされていない FCB の先頭アドレス
FCB のカレントブロック	読み出すブロック番号

戻り値

A	00H	ファイルの作成が成功
	FFH	ファイルの作成が失敗

解　説

FCB 中のレコードサイズ、カレントレコード、ランダムレコードの各フィールドは、このファンクションコールの実行後に必要に応じて設定しなければなりません。

このファンクションコールが失敗するのは、ファイル名が正しくセットされていないか、ディレクトリに空きがないときです。

ファイル名の変更

ファンクション

17H

コール手順

DE	オープンされていないFCBの先頭アドレス。
FCB+0	旧ファイル名
FCB+16	新ファイル名

戻り値

A	00H	ファイル名の変更が成功
	FFH	ファイル名の変更が失敗

解　説

新旧ファイル名に、ワイルドカード文字を使用できます。例えば、旧ファイル名に「????????REL」を、新ファイル名に「????????LIB」を指定すれば、「REL」という拡張子のファイルがすべて、「LIB」という拡張子のファイルに変更されます。

新ファイル名のドライブ番号は無視されます。

ログインベクトルの獲得

ファンクション

18H

コール手順

なし

戻り値

HL	オンラインドライブ情報

解　説

MSXには最大8台のディスクドライブが接続できます。「オンラインドライブ」とは、MSXに実際に接続されているドライブを示します。このファンクションコールを実行すると、各ドライブがオンラインであるかどうかを調べ、結果を表3.7のようにHLレジ

スタに入れて返します。それぞれのビットが1ならば対応するドライブはオンラインであり、0ならば接続されていないことを示します。MSX-DOSのときは、Hレジスタは常に「0」です。

表3.7 ログインベクトル

レジスタ	H								L							
ビット	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ドライブ	常に、0								H:	G:	F:	E:	D:	C:	B:	A:
オンライン・オフライン	各ビットが「1」ならばオンライン、「0」ならばオフライン															

デフォルトドライブ番号の獲得

ファンクション

19H

コール手順

なし

戻り値

A　　デフォルトドライブ番号（A=0、B=1、・・・ H=7）

解　説

現在設定されているドライブ番号を獲得します。

転送先アドレス（DTA）の設定

ファンクション

1AH

コール手順

DE　　設定するDTAアドレス

戻り値

なし

解 説

DTA アドレスは、システムリセット時に 0080H に初期化されますが、このファンクションコールによって、任意のアドレスに設定し直すことができます。

ディスク情報の獲得

ファンクション

1BH

コール手順

E	目的のディスクが入っているドライブ番号

戻り値

A	1クラスタあたりの論理セクタ数
	FFH　指定ドライブがオフライン
BC	セクタのサイズ（バイト単位）
DE	クラスタの総数
HL	未使用クラスタの総数
IX	DPB の先頭アドレス
IY	FAT バッファの先頭アドレス

解 説

指定ドライブのディスクの情報を獲得します。ドライブ番号に「0」を指定すると、デフォルトドライブの指定になります。それ以外は、A ドライブなら「1」、B ドライブなら「2」、・・・を指定します。

このファンクションコールは MSX-DOS 独自のもので、CP/M との互換性はありません。

ランダムな読み出し

ファンクション

21H

コール手順

DE	オープンされたFCBの先頭アドレス
FCBのランダムレコード	読み出すレコード番号

戻り値

A	00H	読み出しが成功
	01H	読み出しが失敗

読み出しが成功したときには、DTA以降の128バイトに読み出したレコードをセットします。

解　説

レコードの大きさは128バイト固定です。

ランダムな書き込み

ファンクション

22H

コール手順

DE	オープンされたFCBの先頭アドレス
FCBのランダムレコード	書き込むレコード番号
DTA以降の128バイト	書き込むデータ

戻り値

A	00H	書き込みが成功
	01H	書き込みが失敗

解　説

レコードの大きさは128バイト固定です。

ファイルサイズの獲得

ファンクション

23H

コール手順

DE　　オープンされたFCBの先頭アドレス

戻り値

A　　00H　　ファイルサイズの獲得が成功

　　　FFH　　ファイルサイズの獲得が失敗

　　　獲得が成功すれば、FCBのランダムレコードフィールドに指定されたファイルの128バイト単位のサイズをセットします。

解　説

ファイルのサイズは128バイト単位です。200バイトのファイルであれば「2」が、257バイトのファイルであれば「3」が設定されます。

ランダムレコードフィールドの設定

ファンクション

24H

コール手順

DE	オープンされたFCBの先頭アドレス
FCBのカレントブロック	目的のブロック
FCBのカレントレコード	目的のレコード

戻り値

ランダムレコードフィールドに、指定されたFCBのカレントブロックフィールドとカレントレコードフィールドから計算したカレントレコードポジションを設定します。

ランダムブロック書き込み

ファンクション

26H

コール手順

DE	オープンされたFCBの先頭アドレス
FCBのレコードサイズ	書き込むレコードサイズ
FCBのランダムレコード	書き込みを開始するレコード
HL	書き込むレコード数
DTA以降のメモリ領域	書き込むデータ

戻り値

A	00H	書き込みが成功
	01H	書き込みが失敗

機　能

書き込みが終わると、ランダムレコードフィールドの値が自動的に更新されて、最後に書き込んだレコードの次のレコードを指します。1つのレコードの大きさはFCBのレコードサイズフィールドによって1バイトから65535バイトまで任意に設定できます。このファンクションコールはMSX-DOS独自のもので、CP/Mとの互換性はありません。

ランダムブロック読み出し

ファンクション

27H

コール手順

DE	オープンされたFCBの先頭アドレス
FCBのレコードサイズ	読み出すレコードサイズ
FCBのランダムレコード	読み出しを開始するレコード
HL	読み出すレコード数

戻り値

A	00H	読み出しが成功
	01H	読み出しが失敗
HL	実際に読み込んだレコードの個数	

解　説

データの読み込みが終わると、ランダムレコードフィールドの値が自動的に更新されます。このファンクションコール実行後、HLレジスタには実際に読み込んだレコードの総数が設定されます。つまり、指定した数のレコードを読み込み終わる前にファイルの終りに達してしまったときには、それまでに読み込んだ実際のレコード数がHLレジスタに入ることになります。

このファンクションコールはMSX-DOS独自のもので、CP/Mとの互換性はありません。

「0」の書き込みをともなうランダムな書き込み

ファンクション

28H

コール手順

DE	オープンされたFCBの先頭アドレス
FCBのランダムレコード	書き込むレコード番号
DTA以降の128バイト	書き込むレコード

戻り値

A	00H	書き込みが成功
	01H	書き込みが失敗

解　説

レコードの大きさは128バイト固定です。

このファンクションコールでは、ファイルサイズより大きなレコード番号が指定されたとき、ファイルの終りの次のレコードから指定されたレコードの直前までのレコードを「0」で埋めた後、指定されたレコードに書き込みます。「0」で埋める点を除けば、ファンクション22H（ランダムな書込み）と同じです。

日付の獲得

ファンクション

2AH

コール手順

なし

戻り値

HL	年	(1980〜2079)
D	月	(1〜12)
E	日	(1〜31)
A	曜日	(0 =日曜、1 =月曜、・・・、6 =土曜)

機　能

クロック IC に設定されている日付のデータを獲得します。

このファンクションコールは MSX-DOS 独自のもので、CP/M との互換性はありません。

日付の設定

ファンクション

2BH

コール手順

HL	年 (1980〜2079)
D	月 (1〜12)
E	日 (1〜31)

戻り値

A	00H	日付の設定が成功
	FFH	日付の設定が失敗

機　能

クロック IC に指定した日付を設定します。

このファンクションコールは MSX-DOS 独自のもので、CP/M との互換性はありません。

時刻の獲得

ファンクション

2CH

コール手順

なし

戻り値

H	時
L	分
D	秒
E	1/100秒

機　能

クロックICに設定されている時刻のデータを獲得します。

このファンクションコールはMSX-DOS独自のもので、CP/Mとの互換性はありません。

時刻の設定

ファンクション

2DH

コール手順

H	時
L	分
D	秒

戻り値

A	00H	時刻の設定が成功
	FFH	時刻の設定が失敗

機　能

クロック IC に時刻のデータを設定します。

このファンクションコールは MSX-DOS 独自のもので、CP/M との互換性はありません。

ベリファイフラグの設定

ファンクション

2EH

コール手順

E	00H	ベリファイフラグをリセット
	01H	ベリファイフラグをセット

戻り値

なし

機　能

ベリファイフラグをセットすると、以降のディスクに対する書き込みをベリファイ付きで行います。ベリファイとは、書き込んだあとでその内容をディスクから読み込み、書き込むべき内容と比較して等しいかどうかをチェックする機能です。

ベリファイ機能はオプションであり、ドライブによってはその機能がないものもあります。その場合、ベリファイフラグを設定しても意味はありません。

このファンクションコールは MSX-DOS 独自のもので、CP/M との互換性はありません。

論理セクタの読み出し

ファンクション

2FH

コール手順

DE	読み出す論理セクタの番号（複数のときはその先頭の論理セクタ番号）
H	読み出す論理セクタの個数

L　　読み出すディスクのドライブ番号（A＝0、B＝1、・・・、H＝7）

戻り値

DTA以降の（論理セクタサイズ×論理セクタの個数）バイトに読み込んだ内容を設定

機　能

指定ドライブの指定論理セクタから指定された個数の論理セクタを読み出し、その内容をDTA以降のメモリに設定します。

このファンクションコールはMSX-DOS独自のもので、CP/Mとの互換性はありません。

論理セクタの書き込み

ファンクション

30H

コール手順

DE　　書き込む論理セクタの番号（複数のときはその先頭の論理セクタ番号）

H　　書き込む論理セクタの個数

L　　書き込むディスクのドライブ番号（A＝0、B＝1、・・・、H＝7）

DTA以降の（論理セクタサイズ×論理セクタの個数）バイトに書き込むデータを設定

戻り値

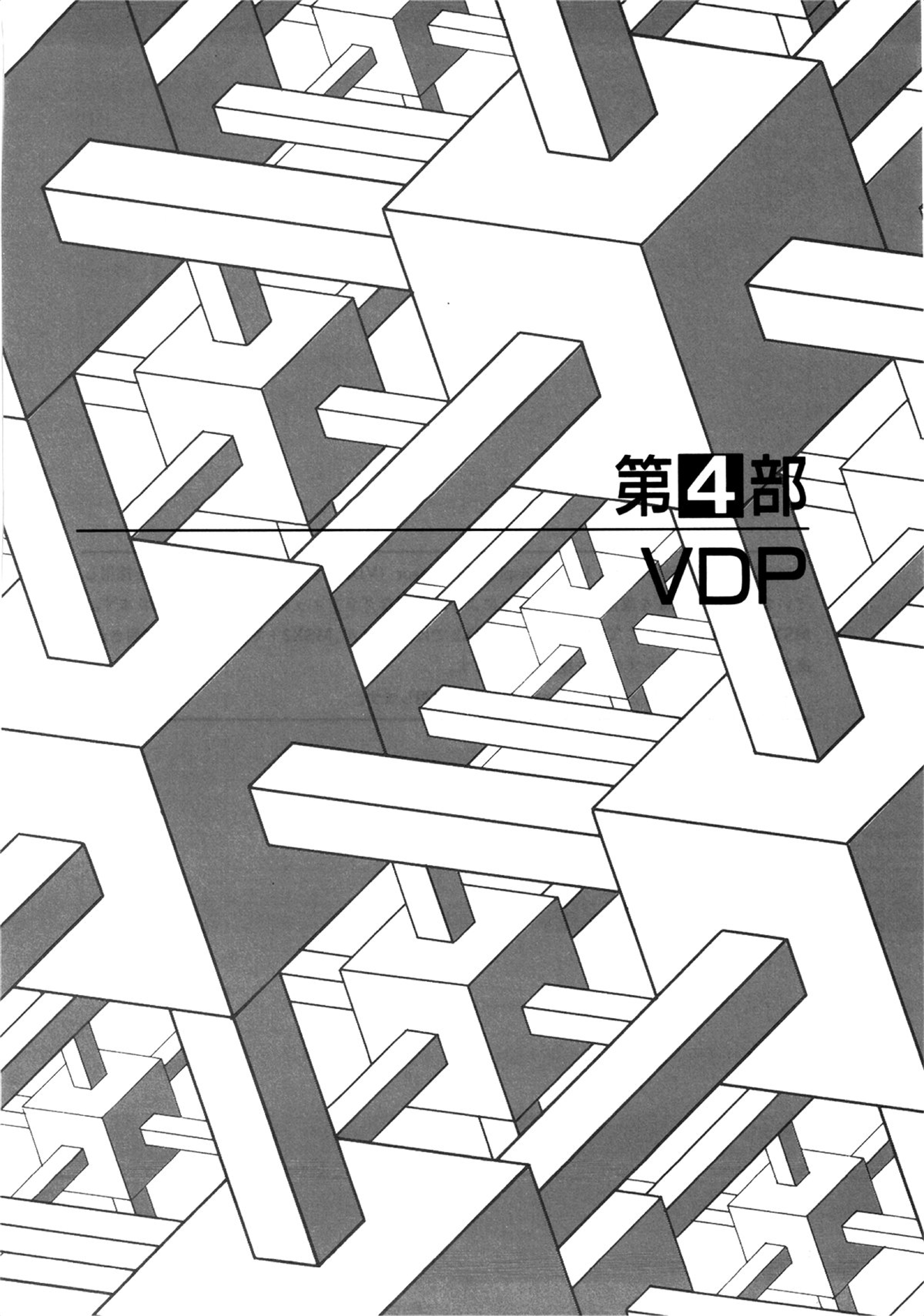
なし

機　能

DTA以降に設定されたデータを指定ドライブの指定論理セクタへ、指定された論理セクタの個数だけ書き込みます。

このファンクションコールはMSX-DOS独自のもので、CP/Mとの互換性はありません。

第4部
VDP



MSX は画面表示のために Video Display Processor（VDP）と呼ばれる専用の LSI を採用しています。この LSI を直接操作することにより、高速なグラフィック表示が可能となります。MSX1 では VDP として TMS9918A が、MSX2 では V9938 が、MSX2＋では V9958 が使用され、高度なグラフィックシステムを提供しています。

第 4 部では、V9938 と V9958 について詳しく説明します。

1章 V9938の構成

「V9938」(MSX-VIDEO) は、MSX2 用の VDP (Video Display Processor) として開発された VLSI です。MSX2 の開発にあたっては、TMS9918A (MSX1 の VDP) とソフトウェア上位互換性があり、さらに機能強化されたビデオプロセッサが必要でした。V9938 はこの条件を満たす VDP として、株式会社アスキーとヤマハ株式会社によって共同開発され、以下の特徴があります。

- ■フルビットマップ表示モード
- ■テキスト 80 文字表示
- ■ X、Y 座標アクセスを実現し、I/O ドライバの負荷軽減と画面モードからの独立が可能
- ■基本コマンドのハードウェア実行により、I/O ドライバの処理の軽減と高性能を実現
- ■デジタイズと外部同期機能の実現
- ■カラーパレット (9 ビット×16 種) とアナログ RGB 出力をサポート
- ■ 1 水平ライン上の表示スプライト数の増加
- ■ハードウェアによる垂直スクロール

1.1 レジスタ

V9938 では、その機能を利用するために内部に「VDP レジスタ」を備えています。特に断りのない場合、これを単にレジスタと呼びます。レジスタは機能によって、

1. コントロールレジスタ
2. ステータスレジスタ
3. パレットレジスタ

に分けられます。この内の、コントロールレジスタとステータスレジスタについては、BASIC の VDP (n) というシステム変数によって参照することができます。

1.1.1 コントロールレジスタ(R#0〜R#23、R#32〜R#46)

V9938の動作を制御する書き込み専用の8ビットレジスタ群です。一般にR#nの記号で表します。R#0〜R#23は主に画面モードの設定に使用し、R#32〜R#46はVDPコマンドの実行時に使用します。V9938にはR#24〜R#31に相当するコントロールレジスタはありません。

1.1.2 ステータスレジスタ(S#0〜S#9)

V9938から得られる各種の情報を読みとるための、読み出し専用の8ビットレジスタ群です。一般にS#nという記号で表します。

1.1.3 パレットレジスタ(P#0〜P#15)

カラーパレットを設定するための書き込み専用のレジスタ群です。一般にP#nという記号で表します。nはパレット番号を意味し、各パレットを512色中の1色に設定します。1つのパレットレジスタは9ビット長になっており、RGB各色につき、それぞれ3ビットずつ使用します。

カラーパレットは、リセット時には表4.1のように設定されます。「0」は輝度が0であることを意味し、「7」は輝度が最も高いことを意味しています。

ただし、GRAPHIC 7*モードではカラーパレットを使わないため、この表は意味を持ちません。また、GRAPHIC 5*モードではカラーコードが0〜3の4種類しか存在しないため、表4.1の4〜Fは意味を持ちません。

注意　* このGRAPHICモードは、BASICのスクリーンモードとは異なります。
詳しくは、「4章 V9938の画面モード」を参照して下さい。

表 4.1　リセット時のカラーパレット設定値

カラーコード	G	R	B
0	0	0	0
1	0	0	0
2	6	1	1
3	7	3	3
4	1	1	7
5	3	2	7
6	1	5	1
7	6	2	7
8	1	7	1
9	3	7	3
A	6	6	1
B	6	6	4
C	4	1	1
D	2	6	5
E	5	5	5
F	7	7	7

1.2 VRAM

V9938には128KバイトのVRAM（VIDEO RAM）と64Kバイトの拡張RAMを接続できます。V9938はこの128Kバイトのアドレス空間をアクセスするために17ビットのアドレスカウンタを持っています。なお、このメモリはV9938によって管理されるものですから、CPUから直接アクセスすることはできません。

拡張RAMはVRAMのように、その内容を画面に表示することはできませんが、VDPコマンド実行時には、ワークエリアとして、VRAMと同等に扱うことができます。ただし、MSX2では拡張RAMは使用しません。

1.3 I/O ポート

V9938はCPUとデータのやり取りを行うために、4つのI/Oポートを持っています。このポートは表4.2の機能を持ち、CPUのI/Oアドレスを通して接続されています。表中m、nで表されたアドレスはMAIN ROMの6、7番地に記録されています。アプリケーションプログラムは、この番地の内容を参照してVDPをアクセスして下さい。

表4.2　V9938のポート

ポート	アドレス	用途
ポート#0 (READ)	m	VRAMからのデータを読み出し
ポート#0 (WRITE)	n	VRAMへのデータ書き込み
ポート#1 (READ)	m+1	ステータスレジスタの読み出し
ポート#1 (WRITE)	n+1	コントロールレジスタへの書き込み
ポート#2 (READ)	n+2	パレットレジスタへの書き込み
ポート#3 (WRITE)	n+3	間接指定されたレジスタへのデータ書き込み

注意　mの値はMAIN ROMの6番地を参照して得る。
nの値はMAIN ROMの7番地を参照して得る。

MSX BASICやBIOSではVDPの書き込み専用レジスタの値を参照する必要が生じたときのために、それらのレジスタに書き込む値をメインメモリのワークエリアやVRAMの空き領域に保存しています。

ユーザープログラムが、この保存されている値を更新しないで、書き込み専用レジスタの値を変更したときには、その後のBASICやBIOSの動作は保証されません。

2章 基本入出力

この章では、VDP を操作するのに必要なコントロールレジスタ、パレットレジスタ、ステータスレジスタ、VDP にアクセスする方法などについて説明します。

2.1 コントロールレジスタのアクセス

V9938 のコントロールレジスタ（R＃0〜R＃46）にデータをセットする方法は、直接指定と間接指定の 2 とおりあります。

2.1.1 直接指定

ポート＃1 に、データ、レジスタ番号の順に出力する方法です。この順番は必ず守らねばならないので、割り込みルーチンなどで V9938 をアクセスしているときには注意が必要です。

	7	6	5	4	3	2	1	0	
Port＃1 1st byte	D7	D6	D5	D4	D3	D2	D1	D0	1. ポート＃1 にデータを出力
Port＃1 2nd byte	1	0	R5	R4	R3	R2	R1	R0	2. ポート＃1 にレジスタ番号を出力 （上位 2 ビットは常に「10」）

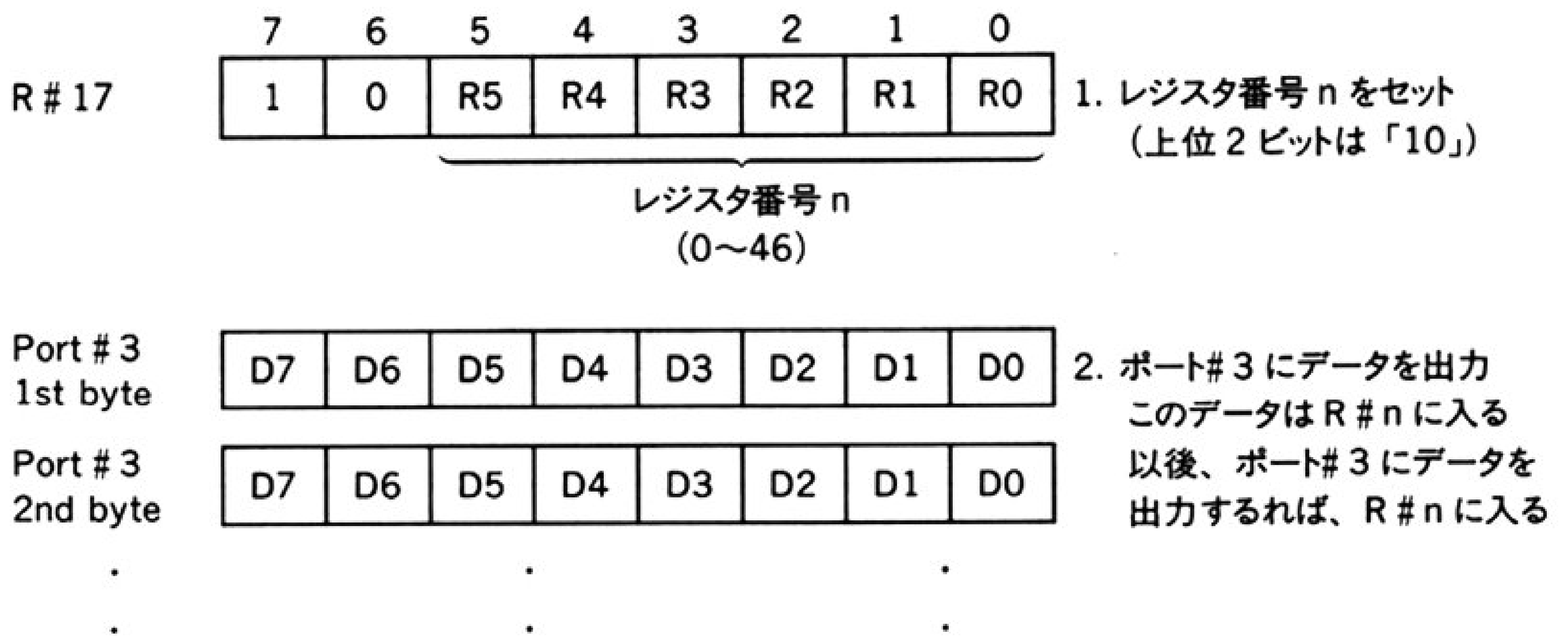
2.1.2 間接指定

コントロールレジスタ R＃17（Control register pointer）でレジスタ番号を指定する方法で、非オートインクリメントとオートインクリメントの 2 つのモードがあります。

1. 非オートインクリメントモード

直接指定でR#17にレジスタ番号をセットし、ポート#3にデータを出力します。R#17の値は間接指定では変更できません。

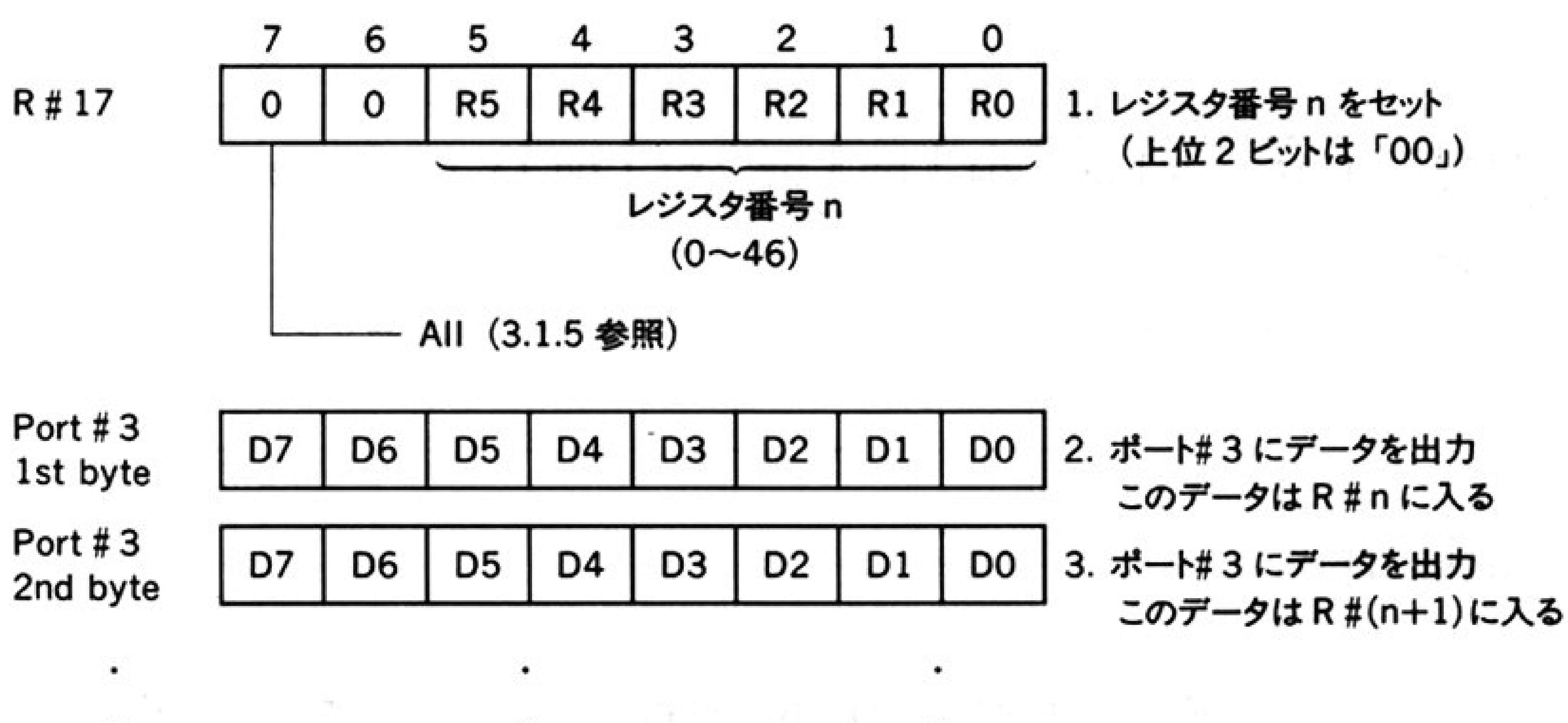
非オートインクリメントモードでは、R#17の内容は不変なので、再セットする必要はありません。このモードはVDPコマンドの実行時など、同じレジスタに連続してデータを送るときに使用します。



2. オートインクリメントモード

直接指定でR#17にレジスタ番号をセットし、ポート#3にデータを出力します。このときR#17のAIIビットに「0」をセットすると、ポート#3にデータを出力するたび、R#17の内容はオートインクリメントされます。

このモードを利用すると、スクリーンモードの変更時など、連続した多数のレジスタを一度に書き換える場合に便利です。



2.2 パレットレジスタのアクセス

パレットレジスタにデータをセットするためには、レジスタR#16 (Color palette address pointer)にパレット番号をセットします。そして、ポート#2にデータを2バイト連続で出力します。これは、V9938のパレットレジスタ(P#0〜P#15)は9ビットの長さを持つため、最初に赤と青の輝度、次に緑の輝度とデータを2度に分けて出力する必要があるからです。

R#16は、ポート#2にデータを2バイト書き込むとオートインクリメントします。

	7	6	5	4	3	2	1	0	
R#16	0	0	0	0	C3	C2	C1	C0	1. パレット番号nをセット
					パレット番号n (0〜15)				
Port#2 1st byte	0	R2	R1	R0	0	B2	B1	B0	2. ポート#2に赤と青の輝度を出力
		赤の輝度 (0〜7)				青の輝度 (0〜7)			
Port#2 2nd byte	0	0	0	0	0	G2	G1	G0	3. 続けて、ポート#2に緑の輝度を出力
						緑の輝度 (0〜7)			

2.3 ステータスレジスタのアクセス

V9938のステータスレジスタ(S#0〜S#9)を読み出すには、レジスタR#15(Status register pointer) にステータスレジスタ番号をセットし、ポート#1を読み出します。

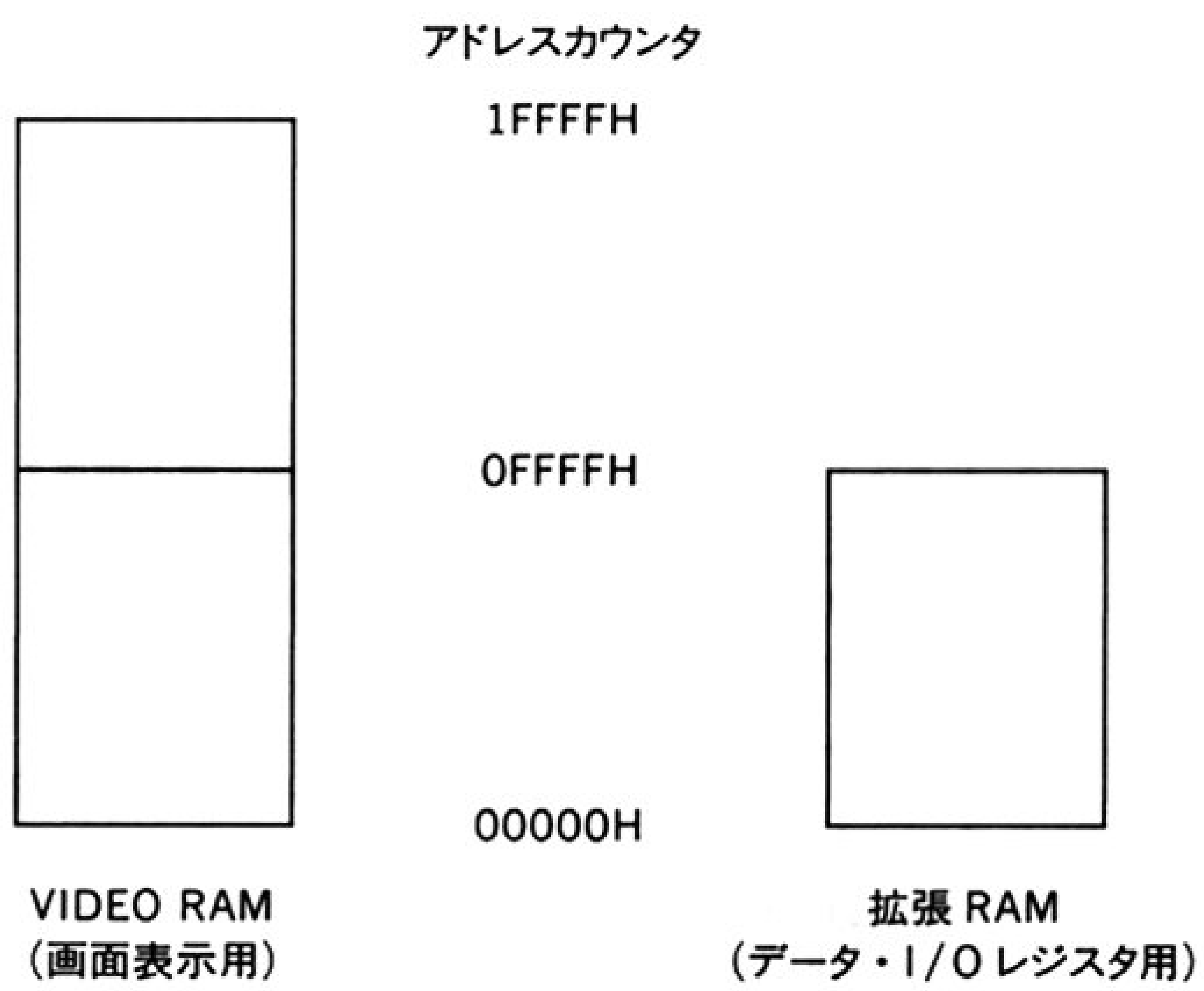
MSXの割り込み処理ルーチンは、R#15に「0」が設定されていることを前提にしています。したがって、ステータスレジスタをアクセスする前には、必ず割り込みを禁止し、目的の処理が終了した後に、R#15に「0」をセットしてから割り込みを解除して下さい。

	7	6	5	4	3	2	1	0	
R#15	0	0	0	0	S3	S2	S1	S0	Status register #
Port#1 Read data	D7	D6	D5	D4	D3	D2	D1	D0	

2.4 VRAM(VIDEO RAM)のアクセス

V9938 には 128K バイトの VRAM と 64K バイトの拡張 RAM を接続することができます。拡張 RAM は VDP コマンド実行時にワークエリアとして使用することができます。ただし、MSX2 では拡張 RAM は使用しないので、R＃45 のビット 6 には常に「0」を設定して下さい。

メモリマップは以下のようになっています。



VRAM アクセスは次の手順で行います。

1. バンク切り換え (VRAM～拡張 RAH)
2. アドレスカウンタセット (A16～A14)
3. アドレスカウンタセット (A7～A0)
4. アドレスカウンタセット (A13～A8) 読み出し・書き込み指定
5. データの読み出し・書き込み

1. アドレスカウンタセット(A16～A14)

レジスタR#14（VRAM Access base address register）を使って、アドレスカウンタの上位3ビット（A16～A14）をセットします。

このレジスタは、TMS9918互換モードのときには、オートインクリメントを指定してもカウントアップしません。

	7	6	5	4	3	2	1	0
R#14	0	0	0	0	0	A16	A15	A14

2. アドレスカウンタセット(A7～A0)

ポート#1に、アドレスカウンタの下位8ビット（A7～A0）にセットするデータを出力します。

	7	6	5	4	3	2	1	0
Port#1	A7	A6	A5	A4	A3	A2	A1	A0

3. アドレスカウンタセット(A13～A8)　読み出し・書き込みの指定

ポート#1に、アドレスカウンタの残り6ビット（A13～A8）にセットするデータと読み出し・書き込みの指定を出力します。

	7	6	5	4	3	2	1	0
Port#1	0	R/W	A13	A12	A11	A10	A9	A8

R/W：
- 0　リード
- 1　ライト

4. データの読み出し・書き込み

ポート#0 にデータを読み出し・書き込みをすると、アドレスカウンタが自動的にインクリメントされるので、連続してアクセスすることができます。

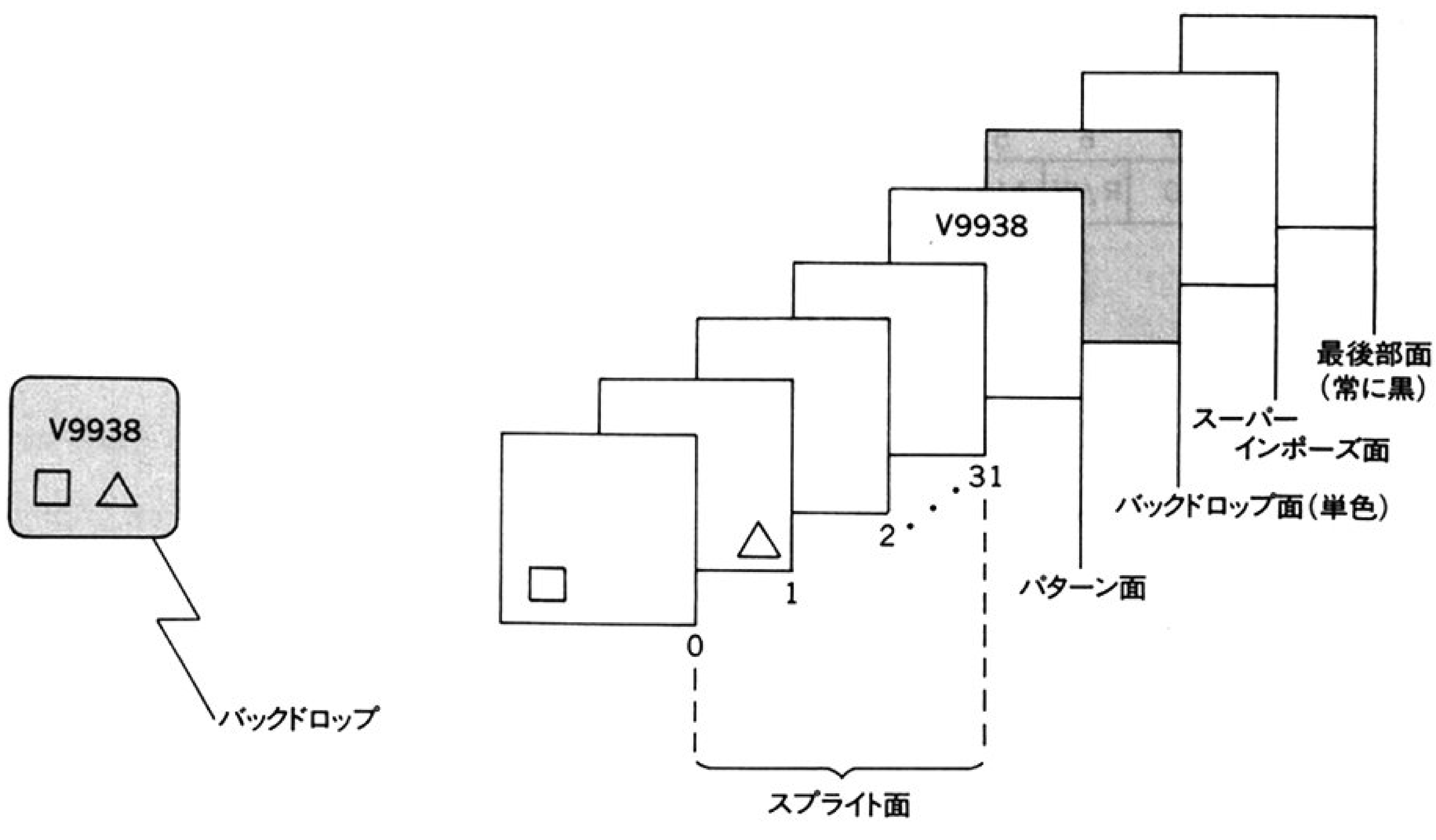
ただし、TMS9918 互換モードのときには、A16〜A14 はインクリメントされません。

	7	6	5	4	3	2	1	0
Port # 0	D7	D6	D5	D4	D3	D2	D1	D0

VRAM のアクセスには、VDP コマンドを使用する方法もあります。これについては5章で詳しく述べます。

2.5 V9938 の画面構成

V9938 の画面は以下のような構成になっています。



3章 レジスタの機能

この章では、VDPを制御するコントロールレジスタ、ステータスレジスタについて説明します。

3.1 コントロールレジスタ#0～#23、#32～#46（Write only）

3.1.1 モードレジスタ

モードレジスタは、各種動作モードを設定するレジスタで、コントロールレジスタ#0、#1、#8、#9に配置されています。

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	M5	M4	M3	0

DG	1のとき、カラーバスを入力モードにして、データをVRAMに取り込む (デジタイズ機能を持ったMSX2でのみ使用可能)
IE2	Interrupt Enable2（1のとき、ライトペンによる割り込みを可能にする）
IE1	Interrupt Enable1（1のとき、水平帰線による割り込みを可能にする）
M5	表示モードの設定に使用する
M4	表示モードの設定に使用する
M3	表示モードの設定に使用する

IE2はライトペン割り込み用のビットです。MSXではこの機能は使用していないので、常に「0」にして下さい。

Mode register 1

R#1

7	6	5	4	3	2	1	0
0	BL	IE0	M1	M2	0	SI	MAG

BL	1=画面表示、0=画面非表示
IE0	Interrupt Enable0(1のとき、垂直帰線による割り込みを可能にする)
M1	表示モードの設定に使用する
M2	表示モードの設定に使用する
SI	スプライトのサイズ　1=16×16、0=8×8
MAG	スプライトの拡大　1=拡大する、0=拡大しない

Mode register 2

R#8

7	6	5	4	3	2	1	0
MS	LP	TP	CB	VR	0	SPD	BW

MS	1=マウスを使用する（カラーバスは入力モード）、0=マウスを使用しない（カラーバスは出力モード）
LP	1=ライトペンを使用する、0=ライトペンを使用しない
TP	カラーコード0の色をカラーパレットの色にする
CB	1=カラーバスを入力モードにする、0=カラーバスを出力モードにする
VR	VRAMの種類を選択する 1=64K×1bitまたは64K×4bit、0=16K×1bitまたは16K×4bit
SPD	1=スプライト非表示、0=スプライト表示
BW	1=白黒32階調、0=カラー（Composit encoderにのみ有効） BWはMSXでは使用していません。

MSはライトペン用の、LPはマウス用のレジスタです。MSXではこの機能は使用していないので、常に「0」にして下さい。

Mode register 3

R#9

7	6	5	4	3	2	1	0
LN	0	S1	S0	IL	EO	NT	DC

LN　1=縦212ドット表示、0=縦192ドット表示
S1　同期モード選択(「7.4 同期モードの選択」参照)
S0　同期モード選択(「7.4 同期モードの選択」参照)
IL　1=Interlace（完全NTSCタイミング)、0=Non Interlace（不完全NTSCタイミング)
EO　1=Even field/Odd fieldで2枚の絵を交互に表示、0=Even field/Odd fieldで同じ絵を表示
NT　1=PAL（313line)、0=NTSC（262line)　RGB出力のみ有効
DC　1=DLCLK端子を入力モードにする、0=DLCLK端子を出力モードにする

3.1.2 テーブルベースアドレスレジスタ

V9938に対してVRAM上の各テーブルの先頭アドレスを宣言するためのレジスタ群です。
表示モードによっては、設定できるデータの値に制限（実際のアドレスと異なる場合）があるので注意して下さい。詳しくは、各表示モードの解説を参照して下さい。

Pattern name table base address register

	7	6	5	4	3	2	1	0
R#2	0	A16	A15	A14	A13	A12	A11	A10

Color table base address register low

	7	6	5	4	3	2	1	0
R#3	A13	A12	A11	A10	A9	A8	A7	A6

Color table base address register high

	7	6	5	4	3	2	1	0
R#10	0	0	0	0	0	A16	A15	A14

Pattern generator table base address register

	7	6	5	4	3	2	1	0
R#4	0	0	A16	A15	A14	A13	A12	A11

Sprite attribute table base address register low

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	A9	1	1

Sprite attribute table base address register high

	7	6	5	4	3	2	1	0
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

3.1.3 カラーレジスタ

V9938の表示色、ブリンクなどを制御するためのレジスタ群です。

Text color / Back drop color register

	7	6	5	4	3	2	1	0	
R#7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0	(GRAPHIC 7 モード時以外)
	BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0	(GRAPHIC 7 モード時)

TC3～TC0　TEXT 1、TEXT 2 モードにおけるテキストの色を指定
BD3～BD0　GRAPHIC 7 以外の表示モードにおけるバックドロップの色を指定
BD7～BD0　GRAPHIC 7 モードにおけるバックドロップの色を指定

Text color / Back color register

	7	6	5	4	3	2	1	0
R#12	T23	T22	T21	T20	BC3	BC2	BC1	BC0

TEXT 2 モードにおいてパターンにブリンクの属性がついているときは、このレジスタで指定された色とR#7で指定された色が交互に表示されます。

T23〜T20　パターンの1の部分の色を指定
BC3〜BC0　パターンの0の部分の色を指定

Blinking period register

	7	6	5	4	3	2	1	0
R#13	ON3	ON2	ON1	ON0	OF3	OF2	OF1	OF0

GRAPHIC 4〜GRAPHIC 7のビットマップモードとTEXT 2モードで、2ページの画面を交互に表示させる(ブリンクさせる)ためのレジスタです。このレジスタにデータをセットし、表示ページを奇数ページにセットするとブリンクを開始します。表4.3を参照して下さい。

ON3〜ON0　偶数ページの表示時間（ビットマップモード時）
　　　　　R#7の色の表示時間（TEXT 2モード時）
OF3〜OF0　奇数ページの表示時間（ビットマップモード時）
　　　　　R#12の色の表示時間（TEXT 2モード時）

Color burst register 1

	7	6	5	4	3	2	1	0
R#20	0	0	0	0	0	0	0	0

Color burst register 2

	7	6	5	4	3	2	1	0
R#21	0	0	1	1	1	0	1	1

Color burst register 3

	7	6	5	4	3	2	1	0
R#22	0	0	0	0	0	1	0	1

カラーバーストレジスタには、それぞれ上記の値がパワーオン時にプリセットされます。この値をすべて0にするとコンポジットビデオ出力の色成分の信号を消すことができます。R#20〜R#22はMSX2では使用しません。

補　足　MSXのコンポジットビデオ出力は、多くの場合V9938のRGBビデオ出力から外部回路により作成されます。したがって、コンポジット出力に影響のある機能は有効ではないとお考え下さい。

3.1.4 ディスプレイレジスタ

CRT 上の表示位置を制御するレジスタ群です。

Display adjust register

	7	6	5	4	3	2	1	0
R＃18	V3	V2	V1	V0	H3	H2	H1	H0
	縦方向の表示位置 (−8〜+7)				横方向の表示位置 (−8〜+7)			

CRT 上の表示位置を補正するためのレジスタです。

左右の補正(H)	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
	左							中央								右
上下の補正(V)	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
	上							中央								下

Display offset register

	7	6	5	4	3	2	1	0
R＃23	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

表示開始ラインをセットするためのレジスタです。このレジスタの値を変えることによって画面の縦スクロールを行うことができます。ただし、スクロールは256ライン単位で行われので、スプライトテーブルなどは別のページに置かなければなりません。

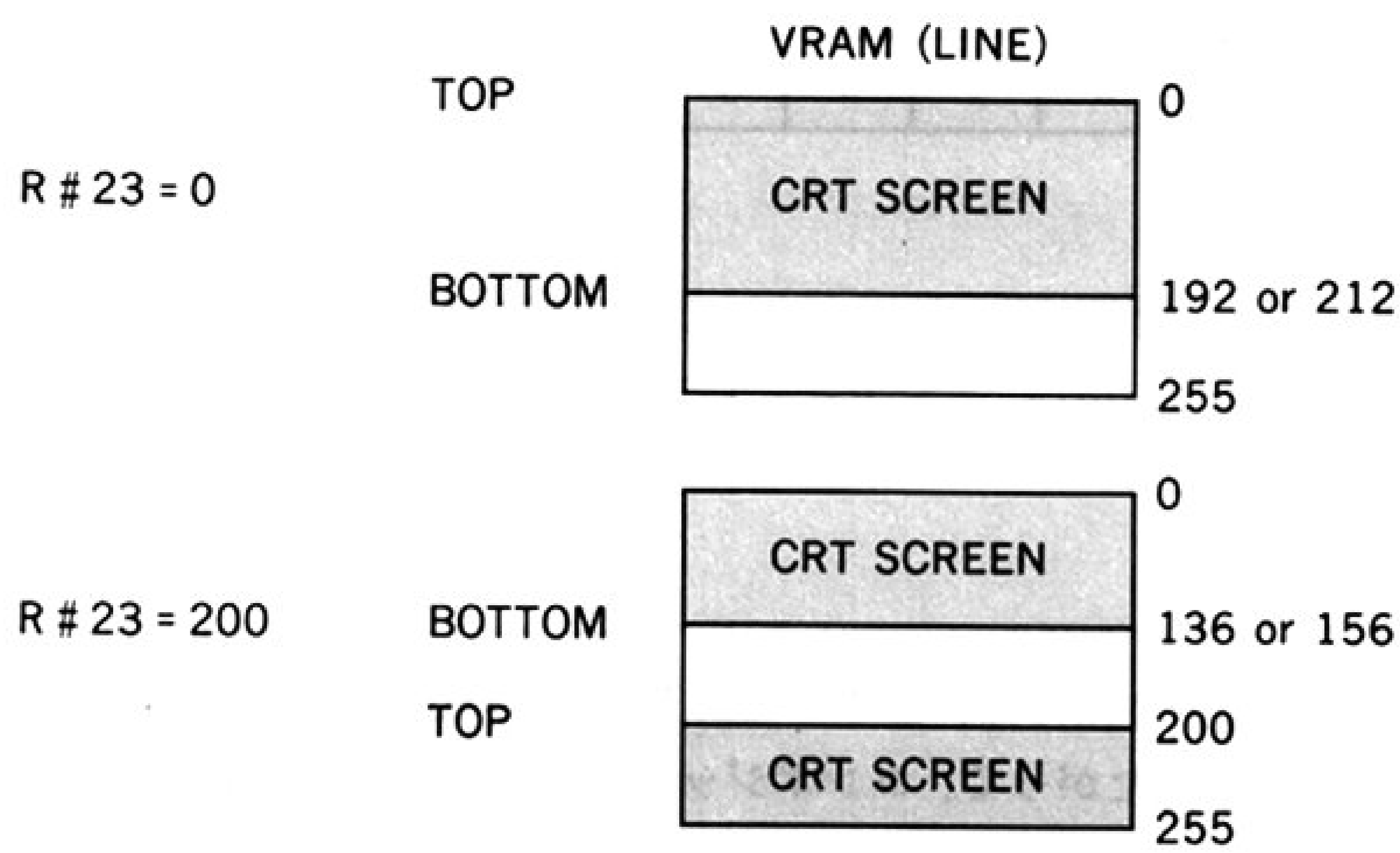


図4.1　ディスプレイオフセットレジスタの設定例

Interrupt line register

	7	6	5	4	3	2	1	0
R#19	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0

V9938 では、CRT が特定の走査線の表示を終えたときに割り込みを発生させることができます。このレジスタに割り込みを発生させる走査線の番号をセットし、R#0 のビット 4 に「1」をセットします。

詳しくは、添付のフロッピーディスクの中に走査線割り込みのサンプルプログラムが入っていますので、参照して下さい。

3.1.5 アクセスレジスタ

V9938 のレジスタや VRAM をアクセスするときに使用するレジスタ群です。

VRAM Access base address register

	7	6	5	4	3	2	1	0
R#14	0	0	0	0	0	A16	A15	A14

V9938 の VRAM をアクセスするときに、アドレスの上位 3 ビットをこのレジスタにセットします。

また、このレジスタの値は、VRAM をアクセスすると A13 からのキャリーを受けて自動的にインクリメントされます。ただし、GRAPHIC 1、GRAPHIC 2、MULTI COLOR、TEXT 1 モードではインクリメントしません。

Status register pointer

	7	6	5	4	3	2	1	0
R#15	0	0	0	0	S3	S2	S1	S0

V9938 のステータスレジスタ（S#0〜S#9）を読み出す際、このレジスタにステータスレジスタの番号（0〜9）をセットします。

Color palette address register

	7	6	5	4	3	2	1	0
R#16	0	0	0	0	C3	C2	C1	C0

V9938のカラーパレットにアクセスする際、このレジスタにパレットの番号をセットします。

Control register pointer

	7	6	5	4	3	2	1	0
R#17	AII	0	R5	R4	R3	R2	R1	R0

R5～R0: レジスタ番号n (0～46)

AII:
0 オートインクリメント
1 オートインクリメント禁止

V9938では、このレジスタの値をポインタとして他のレジスタをアクセスすることができます。また、AIIビットの指定によって、内容を自動的にインクリメントさせることができます。

3.1.6 コマンドレジスタ

V9938のコマンドを実行するときに使用するレジスタ群です。詳しくは5章を参照して下さい。

Source X low register

	7	6	5	4	3	2	1	0
R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0

Source X high register

	7	6	5	4	3	2	1	0
R#33	0	0	0	0	0	0	0	SX8

Source Y low register

	7	6	5	4	3	2	1	0
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0

Source Y high register

	7	6	5	4	3	2	1	0
R # 35	0	0	0	0	0	0	SY9	SY8

Destination X low register

	7	6	5	4	3	2	1	0
R # 36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0

Destination X high register

	7	6	5	4	3	2	1	0
R # 37	0	0	0	0	0	0	0	DX8

Destination Y low register

	7	6	5	4	3	2	1	0
R # 38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0

Destination Y high register

	7	6	5	4	3	2	1	0
R # 39	0	0	0	0	0	0	DY9	DY8

Number of dot X low register

	7	6	5	4	3	2	1	0
R # 40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0

Number of dot X high register

	7	6	5	4	3	2	1	0
R # 41	0	0	0	0	0	0	0	NX8

Number of dot Y low register

	7	6	5	4	3	2	1	0
R # 42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0

Number of dot Y high register

	7	6	5	4	3	2	1	0
R#43	0	0	0	0	0	0	NY9	NY8

Color register

	7	6	5	4	3	2	1	0
R#44	CH3	CH2	CH1	CH0	CL3	CL2	CL1	CL0

Argument register

	7	6	5	4	3	2	1	0
R#45	0	MXC	MXD	MXS	DIY	DIX	EQ	MAJ

Command register

	7	6	5	4	3	2	1	0
R#46	CM3	CM2	CM1	CM0	LO3	LO2	LO1	LO0

3.2 ステータスレジスタ#0〜#9(Read only)

V9938の状態を読み出すための、読み出し専用のレジスタ群です。

Status register 0

	7	6	5	4	3	2	1	0
S#0	F	5S	C	5th sprite #				

F　　垂直帰線割り込みフラグ
　　S#0を読み出すとリセットされる

5S　　第5スプライトフラグ
　　1水平線上にスプライトが5個（GRAPHIC 3〜GRAPHIC 7モードでは9個）並ぶとセットされる

C　　衝突フラグ
　　スプライトが衝突するとセットされる

5th sprite #　第5(第9)スプライトの番号がセットされる

Status register 1

	7	6	5	4	3	2	1	0
S#1	FL	LPS	ID#					FH

FL　ライトペンスイッチ（ライトペンフラグがセットされているとき）
ライトペンが光を検出するとセットされる。このとき、IE2 がセットされていると割り込みを発生する。S#1 を読み出すとリセットされる。
マウススイッチ 2（マウスフラグがセットされているとき）
マウスのスイッチ 2 が押されたらセットされる。S#1 を読み出してもリセットされない。

LPS　ライトペンスイッチ（ライトペンフラグがセットされているとき）
ライトペンのスイッチが押されるとセットされる。S#1 を読み出してもリセットされない。
マウススイッチ 1（マウスフラグがセットされているとき）
マウスのスイッチ 1 が押されたらセットされる。S#1 を読み出してもリセットされない。

ID#　V9938 の ID 番号

FH　水平帰線割り込みフラグ
水平帰線（R#19 で指定）による割り込み（フラグ IE1 がセットされているとき）が発生するとセットされる。S#1 を読み出すとリセットされる。

FL と LPS はライトペン用のレジスタです。MSX ではこのインターフェイスは使用していないので、ビット 6、7 は意味を持ちません。

Status register 2

	7	6	5	4	3	2	1	0
S#2	TR	VR	HR	BD	1	1	EO	CE

TR　転送レディフラグ
CPU to VRAM、VRAM to CPU などのコマンドを実行するときは、CPU はこのフラグを見ながらデータを読み書き(転送)する。1 のとき転送可。

VR　垂直帰線期間フラグ
垂直帰線期間中は 1 になる。

HR　水平帰線期間フラグ
水平帰線期間中は 1 になる。

BD　　境界色発見フラグ

　　サーチコマンドの実行で、境界色または非境界色を発見したら1になる。

EO　　表示フィールドフラグ

　　0=第1フィールド、1=第2フィールド

CE　　コマンド実行フラグ

　　コマンドを実行中は1になる。

Column register low

	7	6	5	4	3	2	1	0
S#3	X7	X6	X5	X4	X3	X2	X1	X0

Column register high

	7	6	5	4	3	2	1	0
S#4	1	1	1	1	1	1	1	X8

Row register low

	7	6	5	4	3	2	1	0
S#5	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

Row register high

	7	6	5	4	3	2	1	0
S#6	1	1	1	1	1	1	EO	Y8

これらのレジスタには、スプライトの衝突座標などがセットされます。詳しくは、「6.2.8 スプライトの衝突」をご参照下さい。

Color register

	7	6	5	4	3	2	1	0
S#7	C7	C6	C5	C4	C3	C2	C1	C0

POINT、VRAM to CPU などのコマンドを実行すると、VRAM のデータがこのレジスタにセットされます。

Border X register low

	7	6	5	4	3	2	1	0
S#8	BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0

Border X register high

	7	6	5	4	3	2	1	0
S#9	1	1	1	1	1	1	1	BX8

サーチコマンドで発見した境界色または非境界色のX座標が、このレジスタにセットされます。

4章
V9938の画面モード

この章では、V9938の各画面モードのレジスタの設定方法などについて説明します。

4.1 TEXT 1（SCREEN 0、40字モード）

4.1.1 特徴

■1パターンのサイズ	横6ドット　縦8ドット
■パターンの数	256種類
■画面上のパターン数	横40パターン　縦24パターン
■パターンの色	512色中2色（全画面）
■1画面表示に必要なVRAM容量	4Kバイト

4.1.2 関係するレジスタとVRAMの領域

■パターンのフォント	VRAM パターンジェネレータテーブル
■画面上のパターンの位置	VRAM パターンネームテーブル
■パターンの1の部分の色コード	R#7　上位4ビット
■パターンの0の部分の色コード	R#7　下位4ビット
■バックドロップの色コード	R#7　下位4ビット

4.1.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	0	0	0	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	1	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	$\overline{NT}$	DC

□は表示モード設定用のビット（M5～M1）をTEXT 1モード（00001）にセットした例です。その他のビットは任意に設定します。MSXのシステムソフトウェアはモードレジスタをそれぞれ00000000、01110000、00001000、00001000の値で初期化します。

2. パターンジェネレータテーブルの設定

パターンジェネレータテーブルは、パターン(文字)のフォントを記憶させるエリアです。

パターンには#0～#255の番号がつけられ、パターンを画面に表示するときには、このパターン番号で指定します。

パターンジェネレータテーブルの先頭アドレスはレジスタR#4にセットします。指定できるのは先頭アドレスの上位6ビット（A16～A11）のみで、下位11ビット（A10～A0）は「0」とみなされます。したがって、パターンジェネレータテーブルの先頭アドレスとして指定できるのは00000Hから2Kバイト単位の位置になります。

1個のパターンのフォントは8バイトで構成され、各バイトの下位2ビットは表示されません。

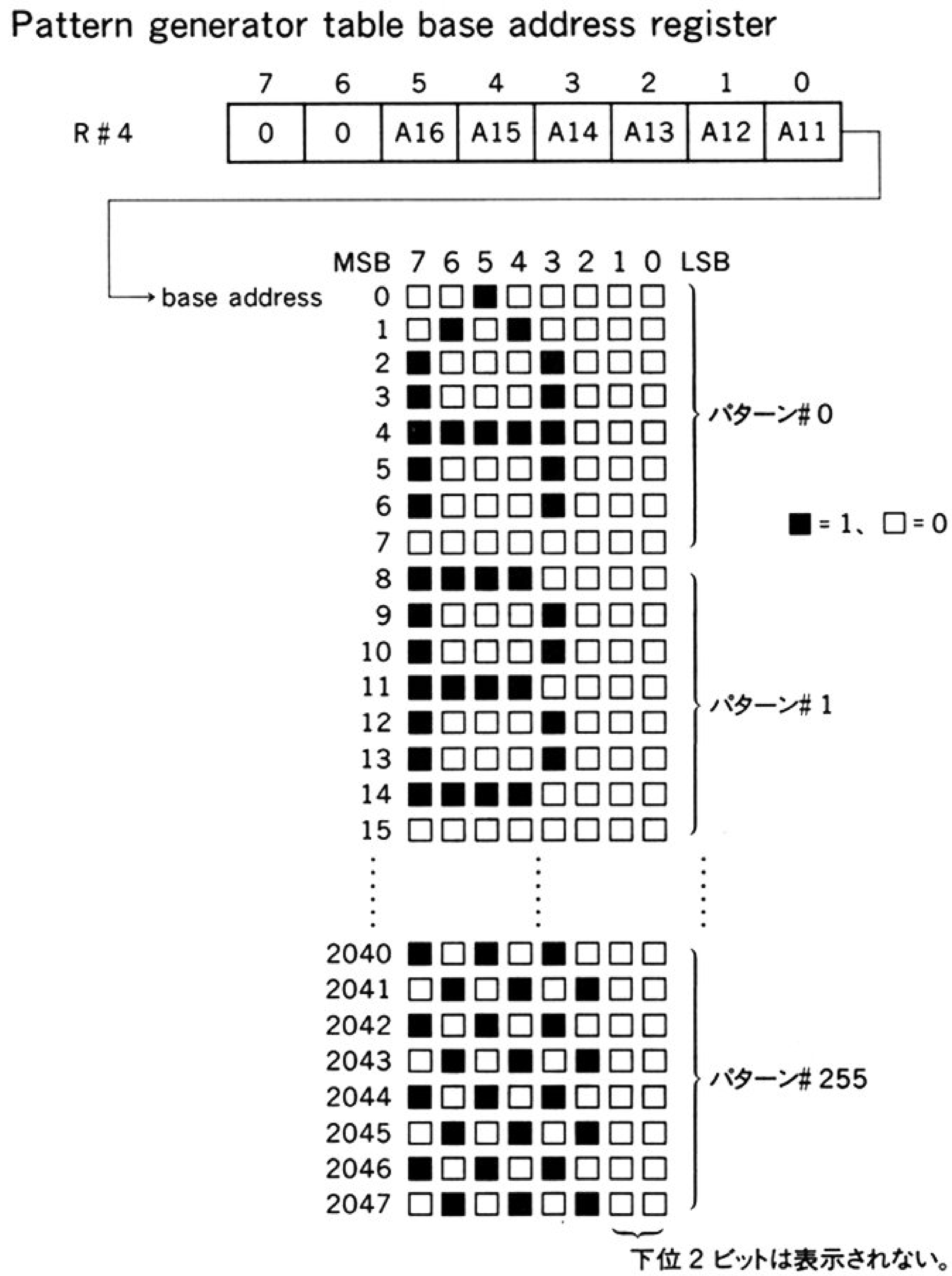


図4.2 TEXT 1モードのパターンジェネレータテーブル例

3. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の1パターンに対応しています。このテーブルに0〜255のパターン番号を書き込むと、対応する画面上の位置に指定したパターンが表示されます。

パターンネームテーブルの先頭アドレスはレジスタR#2にセットします。指定できるのは先頭アドレスの上位6ビット（A16〜A11）のみで、下位11ビット（A10〜A0）は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは00000Hから1Kバイト単位の位置になります。

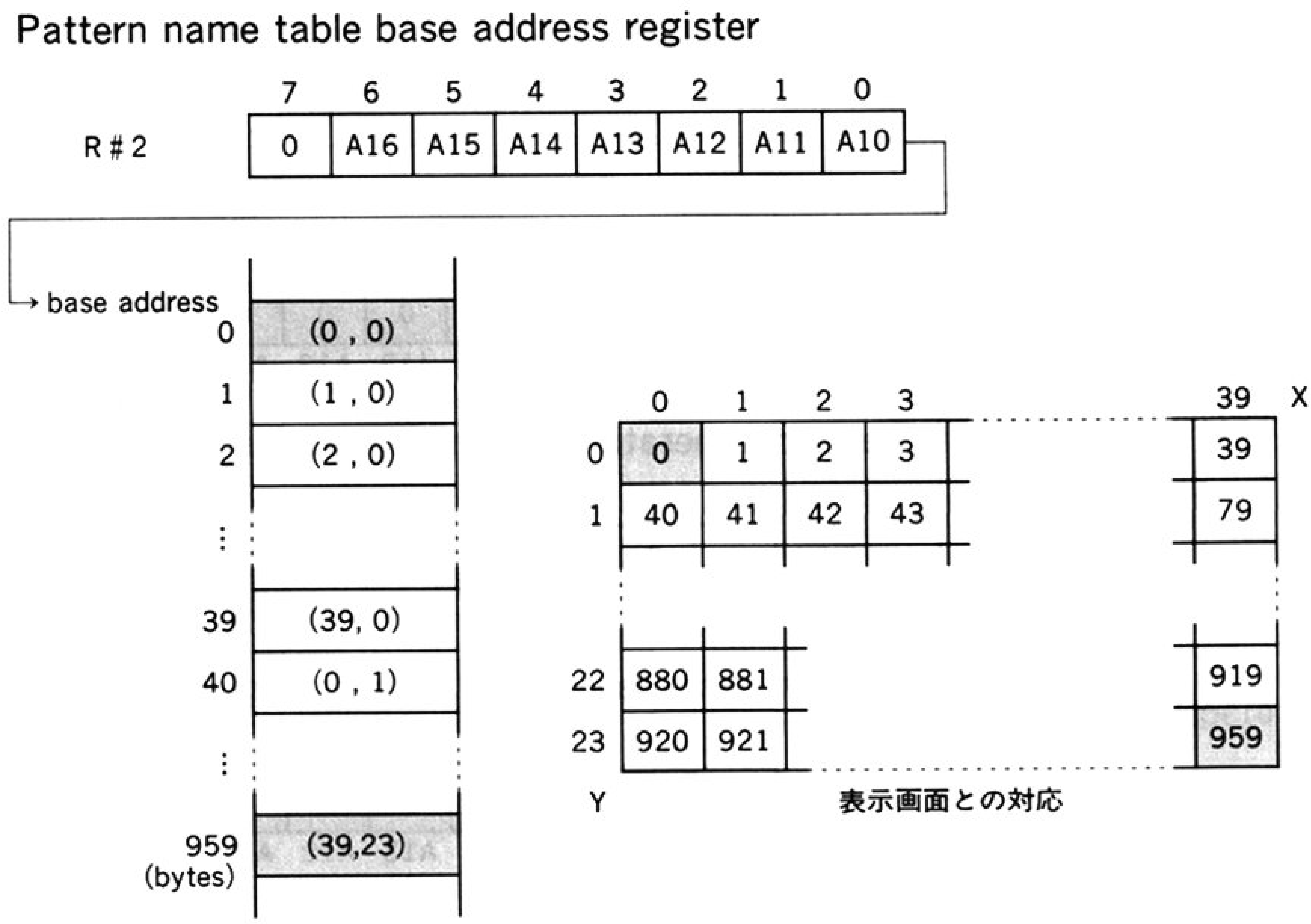
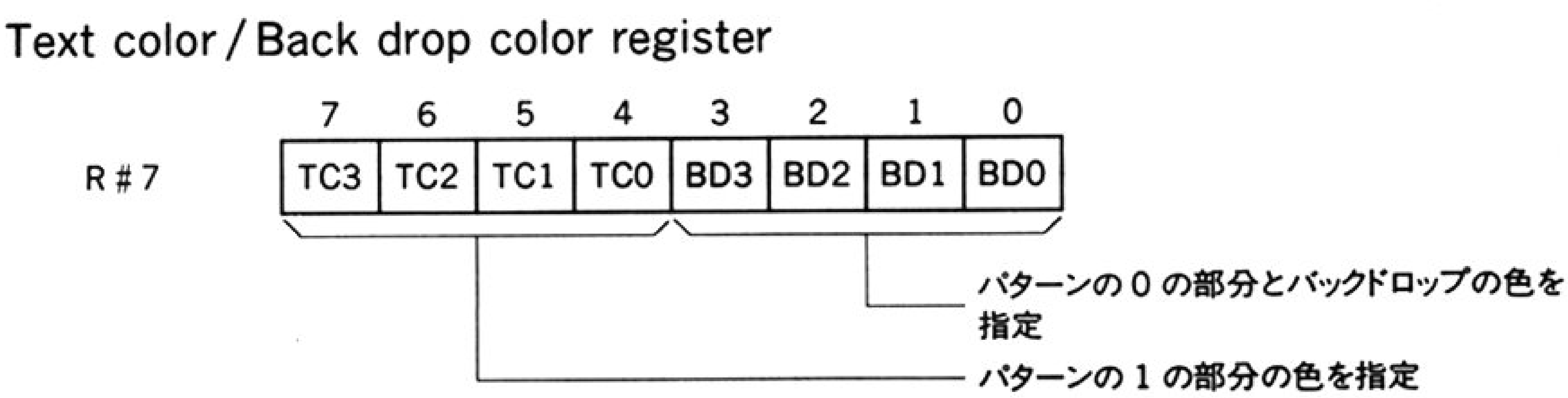


図 4.3　TEXT 1 モードのパターンネームテーブル

4. カラーレジスタの設定

画面の色はR#7で指定します。背景色はR#7の下位4ビット、前景色はR#7の上位4ビットで指定されたパレットによって色が決まります。フォントパターン中の、「0」の部分は背景色、「1」の部分は前景色で表示されます。なお、TEXT1モードでは画面のバックドロップの色を設定することはできず、背景色と同じになります。



4.1.4 TEXT 1 (SCREEN 0、40字)モードのVRAMマップ

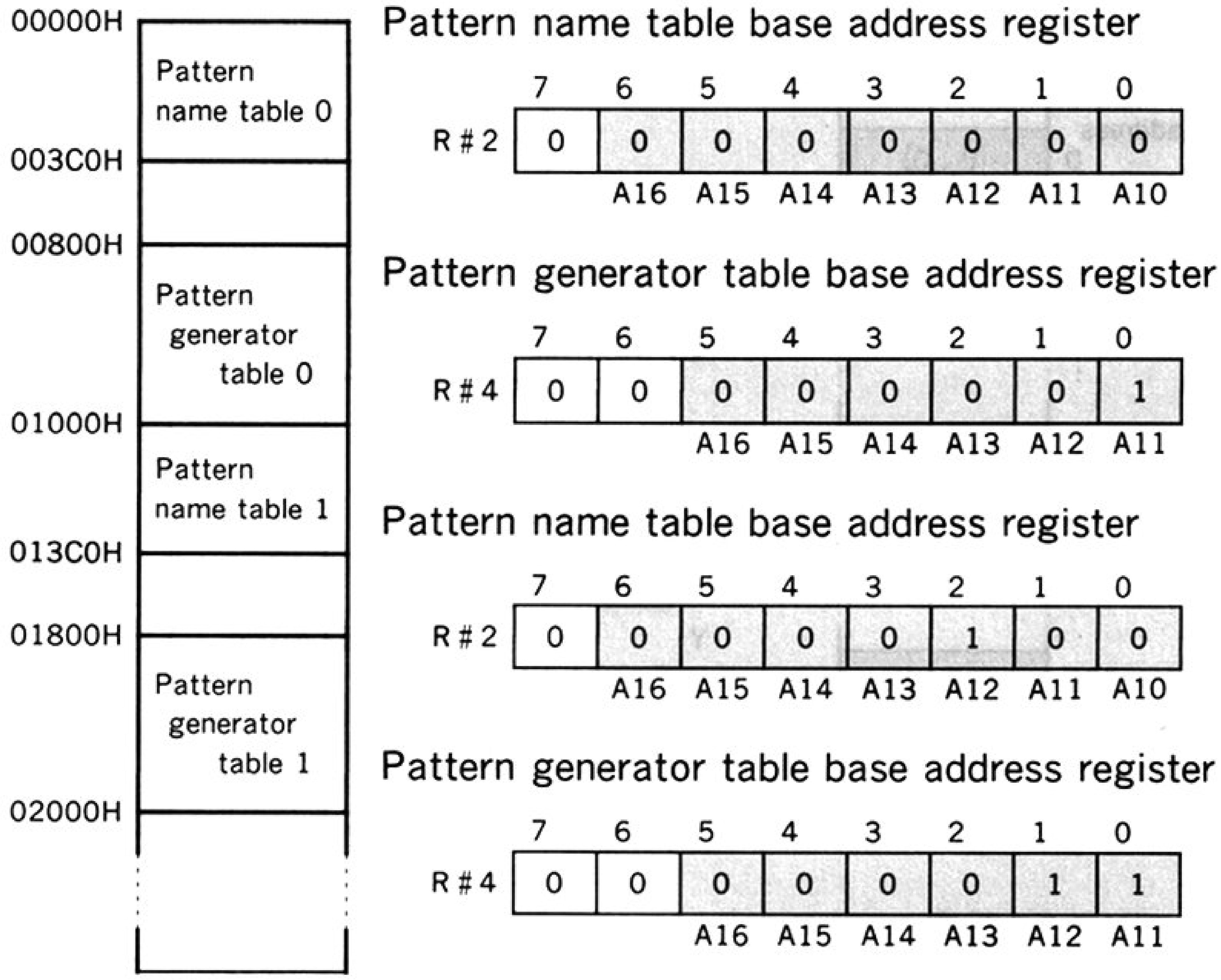


図4.4 TEXT 1モードのVRAMマップ

以下同様に、最高32ページまで割り付けが可能（VRAM 128K実装機種）です。ただしMSX-BASICではこのモードはページ0のみサポートしています。

パレット情報はパレットレジスタが書き込み専用のため、MSXのBIOSが管理し、VRAMの一部を使って記憶しています。この画面モードでは、0400Hから32バイトがパレットテーブルとして使われます（MSXのシステムソフトウェアが設定する）。

color 0の情報は

	7	6	5	4	3	2	1	0
0400H	0	R	R	R	0	B	B	B
0401H	0	0	0	0	0	G	G	G
・				・				・

のように連続2バイトで、RGB各3bitが格納されています。この情報は、パレット制御用のcolor命令で参照されます。

4.2 TEXT 2（SCREEN 0、80字モード）

4.2.1 特徴

■ 1パターンのサイズ	横6ドット　縦8ドット
■パターンの数	256種類
■画面上のパターン数	横80パターン　縦24パターン
	横80パターン　縦26.5パターン
■パターンのブリンク	1文字ごとに指定可能
■パターンの色	512色中2色(全画面)
	ブリンクを使用すると4色
■ 1画面表示に必要なVRAM容量	8Kバイト

4.2.2 関係するレジスタとVRAMの領域

■パターンのフォント	VRAM パターンジェネレータテーブル	
■画面上のパターンの位置	VRAM パターンネームテーブル	
■ブリンクの属性	VRAM カラーテーブル	
■パターンの1の部分の色コード	R#7　上位4ビット	
■パターンの0の部分の色コード	R#7　下位4ビット	
■バックドロップの色コード	R#7　下位4ビット	
■パターンの1の部分の色コード	R#12 上位4ビット	ブリンク用
■パターンの0の部分の色コード	R#12 下位4ビット	ブリンク用

4.2.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	0	1	0	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	1	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	$\overline{NT}$	DC

□は表示モード設定用のビット（M5〜M1）をTEXT 2モード（01001）にセットした例です。

この表示モードでは、LN＝1のとき26.5行、LN＝0のとき24行になります。その他のビットは任意に設定します。MSXのシステムソフトウェアはモードレジスタをそれぞれ00001000、01110000、00001000、00001000の値で初期化します。

2. パターンジェネレータテーブルの設定

TEXT 1モードと同じです。TEXT 1モードのパターンジェネレータテーブルの項を参照して下さい。

3. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の1パターンに対応しています。このテーブルに0〜255のパターン番号を書き込むと、対応する画面上の位置に指定したパターンが表示されます。

画面上に表示されるパターンはLN＝0のとき横80、縦24パターンで、LN＝1のときは横80、縦26.5パターンです。縦の27個目のパターンは上半分のみ表示されます。ただし、BASICではLN＝1のモードはサポートしていません。

パターンネームテーブルの先頭アドレスはレジスタR#2にセットします。指定できるのは、パターンネームテーブルの先頭アドレスの上位5ビット（A16〜A12）のみで、下位12ビット（A11〜A0）は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは00000Hから4Kバイト単位の位置になります。

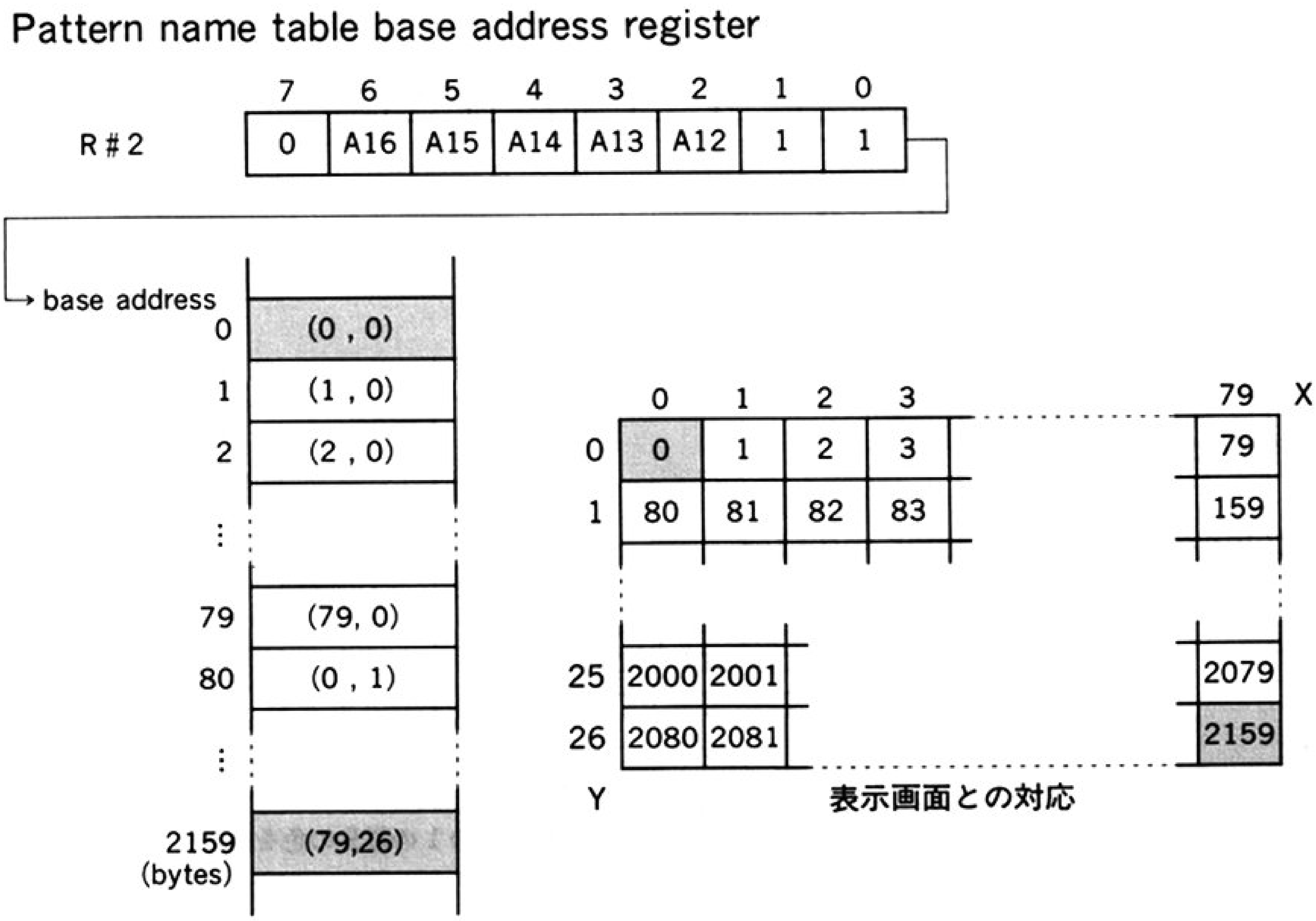


図4.5 TEXT 2モードのパターンネームテーブル

4. カラーテーブルの設定

TEXT 2モードでは、各パターン（文字）に1ビットずつアトリビュートエリアがあり、このビットを1にすることによって、そのパターンにブリンクの属性を持たせることができます。

カラーテーブルの先頭アドレスはレジスタR#3とR#10にセットします。カラーテーブルの先頭アドレスとして指定できるのは00000Hから512バイト単位の位置になります。

Color table base address register

	7	6	5	4	3	2	1	0
R#3	A13	A12	A11	A10	A9	1	1	1
R#10	0	0	0	0	0	A16	A15	A14

→ base address

	7	6	5	4	3	2	1	0
0	(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)
1	(8,0)	(9,0)	(10,0)	(11,0)	(12,0)	(13,0)	(14,0)	(15,0)
⋮								
269	(72,26)							(79,26)

図 4.6　TEXT 2 モードのカラーテーブル

5. カラーレジスタの設定

ブリンクの属性を与えられたパターンは、レジスタ R#7 とレジスタ R#12 で指定された色コードを交互に表示します。

Text color / Back drop color register

	7	6	5	4	3	2	1	0
R#7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0

パターンの 0 の部分とバックドロップの色を指定

パターンの 1 の部分の色を指定

Text color / Back color register

	7	6	5	4	3	2	1	0
R#12	T23	T22	T21	T20	BC3	BC2	BC1	BC0

ブリンク時のパターンの 0 の部分の色を指定

ブリンク時のパターンの 1 の部分の色を指定

6. ブリンクレジスタの設定

ブリンクの属性を与えられたパターンは、レジスタ R#7 とレジスタ R#12 で指定された色コードを交互に表示しますが、この時間の間隔はレジスタ R#13 で指定します。R#13 の上位 4 ビットは本来の文字色が表示されている時間を、下位 4 ビットはブリンク色が表示されている時間を表 4.3 のように設定します。詳しくは表 4.3 を参照して下さい。



図 4.7　ブリンクの周期

Blinking period register



表 4.3　ブリンクのデータと時間一覧

ON TIME と OFF TIME のデータと時間の関係は、このようになります。このデータは、NTSC モードにおいて計算されたものです。

DATA (BIN)	TIME (ms)
0 0 0 0	0
0 0 0 1	166.9
0 0 1 0	333.8
0 0 1 1	500.6
0 1 0 0	667.5
0 1 0 1	834.4
0 1 1 0	1001.3
0 1 1 1	1168.2
1 0 0 0	1335.1
1 0 0 1	1501.9
1 0 1 0	1668.8
1 0 1 1	1835.7
1 1 0 0	2002.6
1 1 0 1	2169.5
1 1 1 0	2336.3
1 1 1 1	2503.2

4.2.4 TEXT 2 (SCREEN 0、80字)モードのVRAMマップ

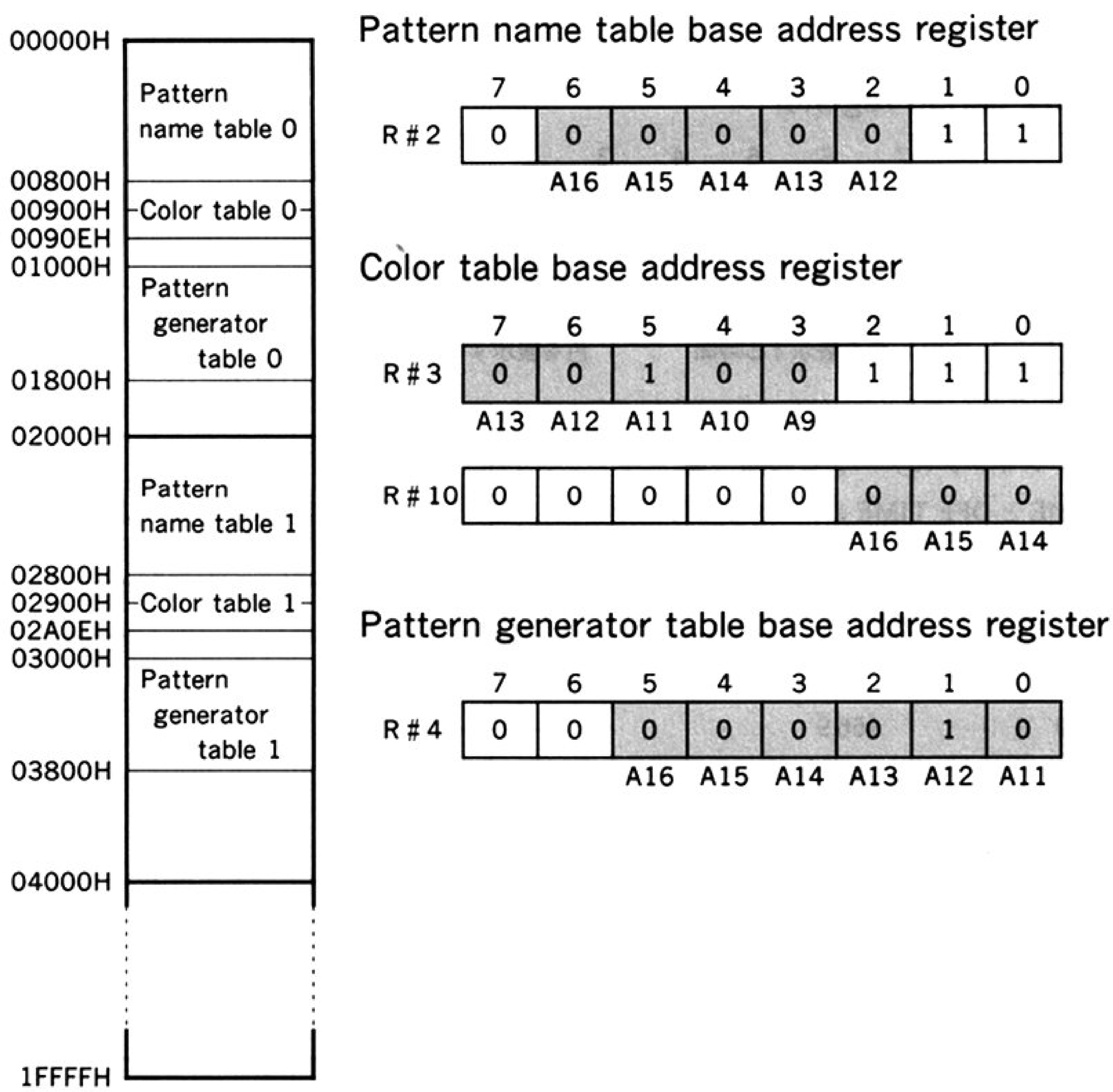


図4.8 TEXT 2モードのVRAMマップ

以下同様に、最高16ページまで割り付けが可能(VRAM 128K実装機種)です。ただし、BASICではこのモードはページ0のみサポートし、カラーテーブル、ブリンク機能はサポートしていません。

この画面モードでは、0F00Hから32バイトがパレットテーブルとして使われます(MSXのシステムソフトウェアが設定する)。

4.3 MULTI COLOR (SCREEN 3)

4.3.1 特徴

■1ブロックのサイズ	横4ドット　縦4ドット
■画面上のブロック数	横64ブロック　縦48ドット
■ブロックの色	512色中16色（各ブロックごとに指定可）
■スプライト	モード1
■1画面表示に必要なVRAM容量	4Kバイト

MULTI COLORモードは他のグラフィックモードとは異なり、1画面が横64個、縦48個のカラーブロックで構成されます。各カラーブロックは、ブロックごとに512色中から16色を指定できます。

4.3.2 関係するレジスタとVRAMの領域

■カラーブロックの色コード	VRAMパターンジェネレータテーブル
■カラーブロックの位置	VRAMパターンネームテーブル
■バックドロップの色コード	R#7　下位4ビット
■スプライト	VRAMスプライトアトリビュートテーブル
	VRAMスプライトパターンテーブル

4.3.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	0	0	0	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	0	1	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

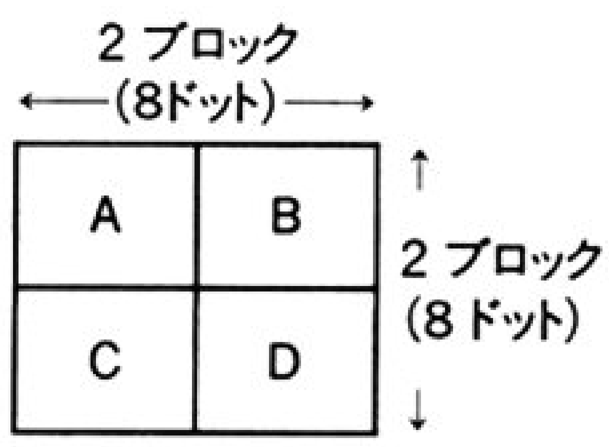
	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	E0	$\overline{NT}$	DC

□は表示モード設定用のビット（M5～M1）をMULTI COLORモード（00010）にセットした例です。その他のビットは任意に設定します。MSXのシステムソフトウェアはモードレジスタをそれぞれ00000000、01101000、00001000、00001000の値で初期化します。

2. パターンジェネレータテーブルの設定

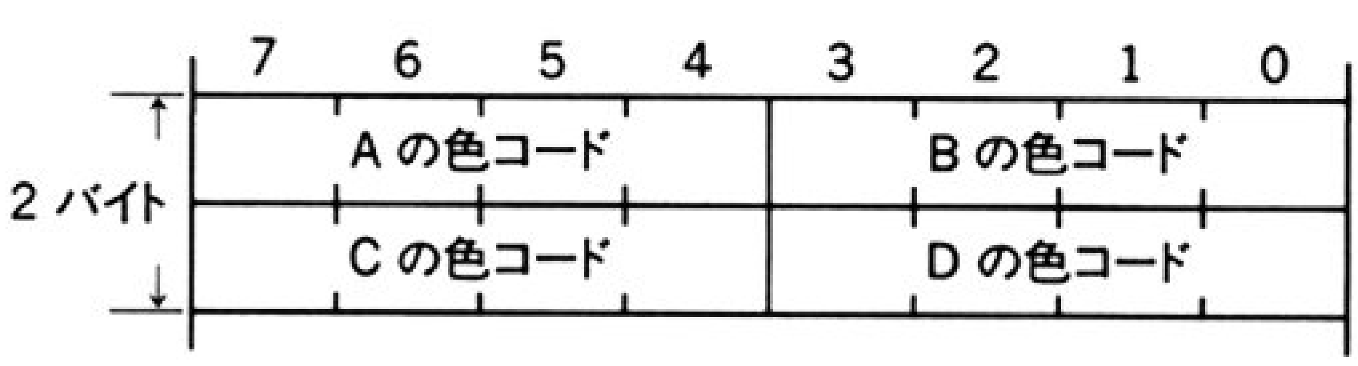
パターンジェネレータテーブルはカラーブロックの色を記憶させるエリアです。

1個のパターンは4個のカラーブロックで構成されます。このパターンは画面の表示領域のドット数を256×192とした場合、8×8に相当する大きさです。



A、B、C、Dの各カラーブロックごとに16色を指定することができます。

1個のパターンに含まれるカラーブロックの色は下図のように2バイトで表現します。



1個のパターン名に対して、表示のY座標によって自動的に選択される4個のパターンが対応します。

パターンジェネレータテーブルの先頭アドレスはレジスタR#4にセットします。指定できるのは、先頭アドレスの上位6ビット（A16～A11）のみで、下位11ビット（A10～A0）は「0」とみなされます。したがって、パターンジェネレータテーブルの先頭アドレスとして指定できるのは00000Hから2Kバイト単位の位置になります。

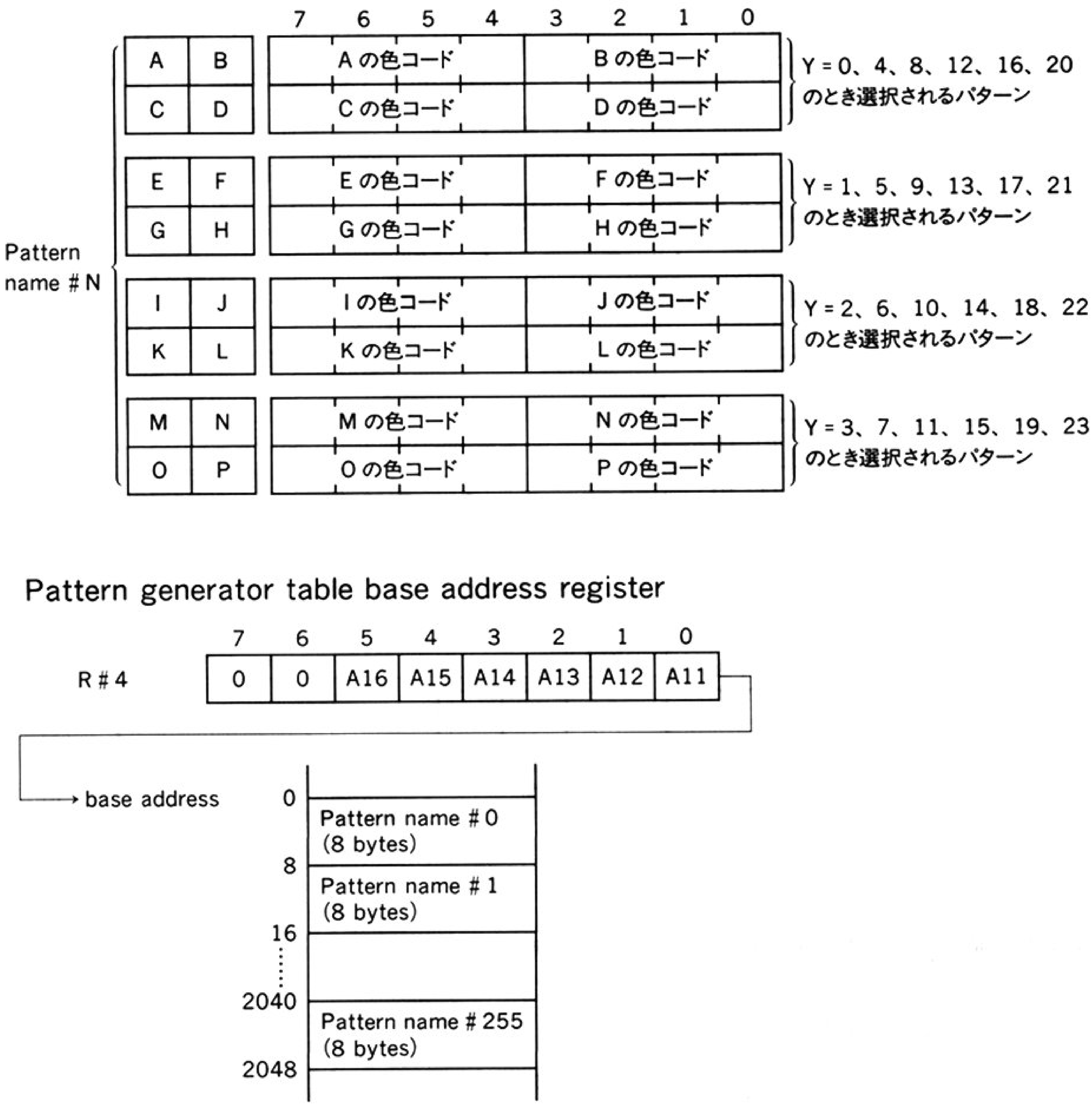


図4.9 MULTI COLORモードのパターンジェネレータテーブル

3. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の1パターンに対応しています。このテーブルに0〜255のパターン番号を書き込むと、対応する画面上の位置に指定したパターンが表示されます。

パターンネームテーブルの先頭アドレスはレジスタR#2にセットします。指定できるのは、パターンネームテーブルの先頭アドレスの上位7ビット（A16〜A10）のみで、下位10ビット（A9〜A0）は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは00000Hから1Kバイト単位の位置になります。

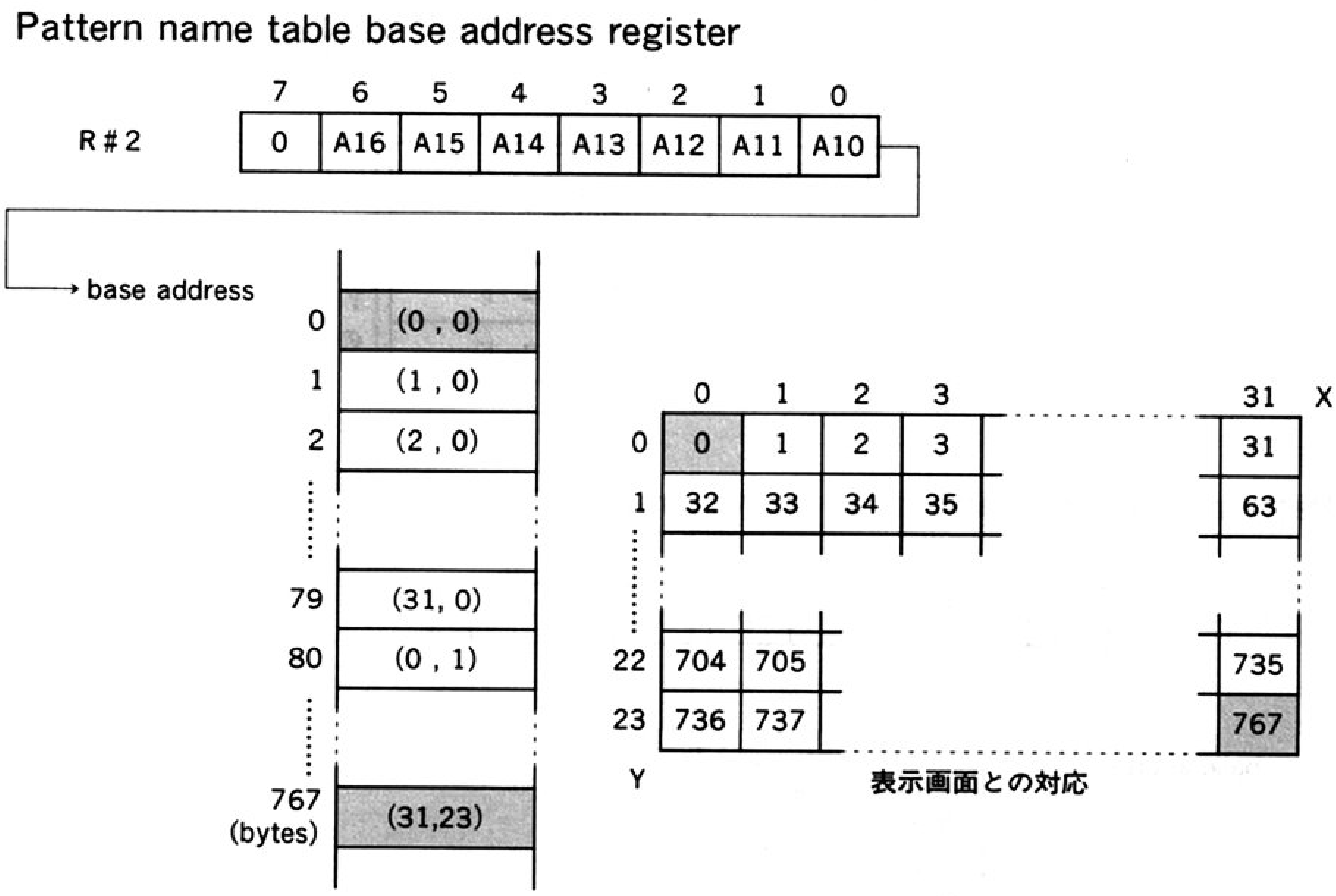
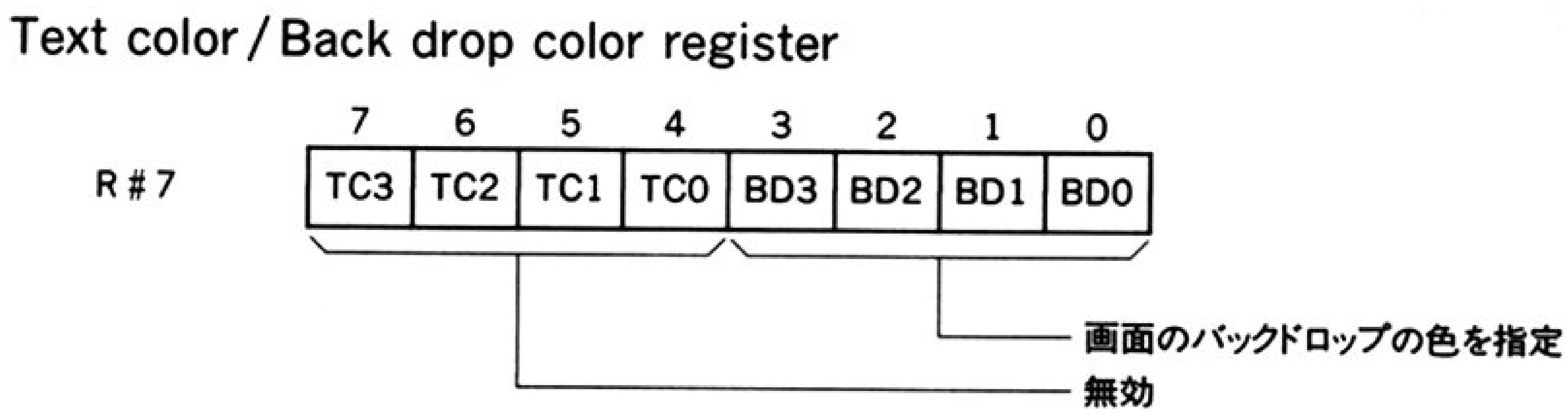


図4.10　MULTI COLOR モードのパターンネームテーブル

4. カラーレジスタの設定

R#7で画面のバックドロップの色が指定できます。



5. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタ R#5 と R#11 に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタ R#6 にセットします。詳しくは、「6.1 スプライトモード 1」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	A9	A8	A7
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.3.4 MULTI COLOR(SCREEN 3)モードのVRAMマップ

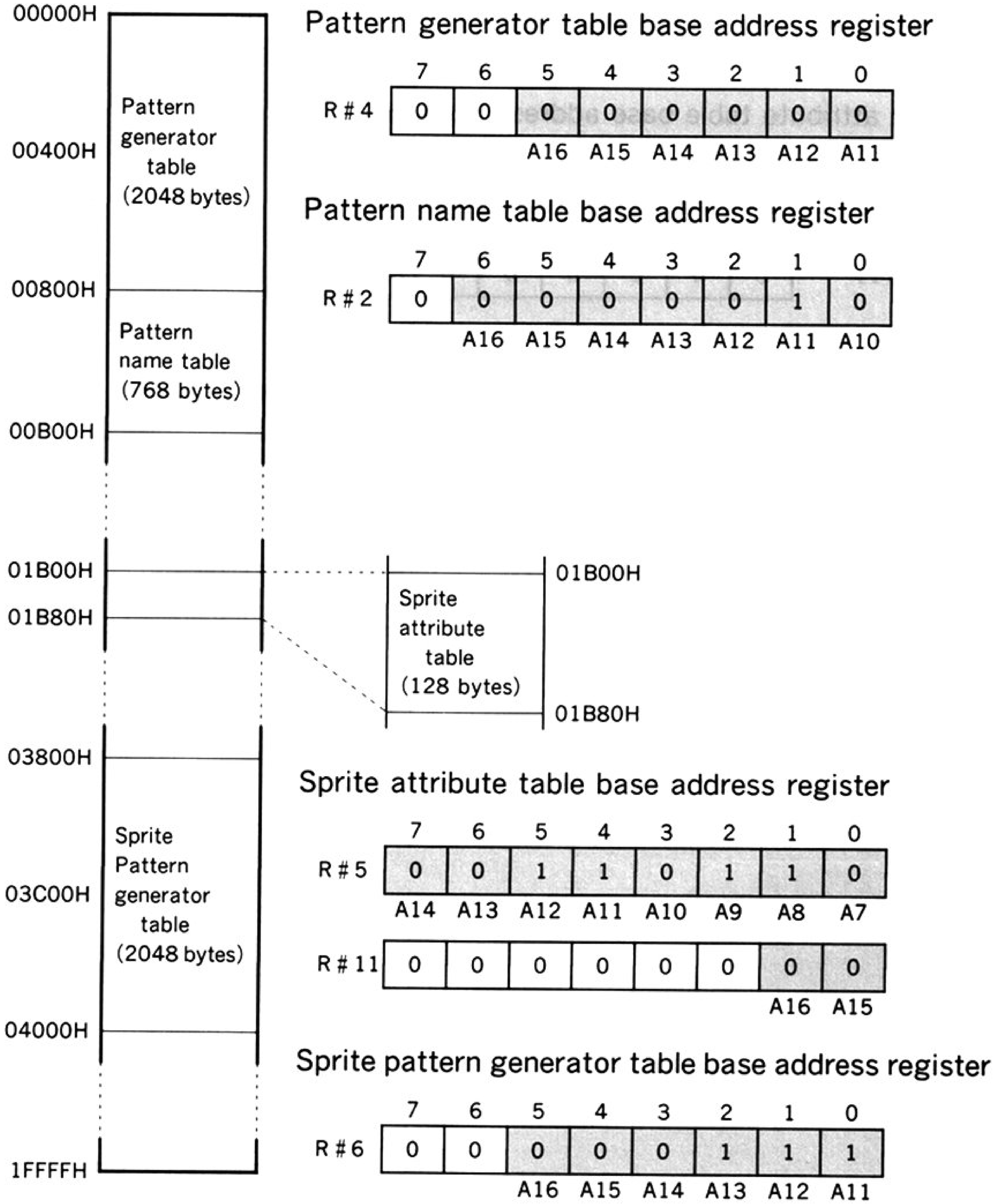


図4.11 MULTI COLOR モードのVRAMマップ

以下同様に8ページまで割り付けが可能（VRAM 128K実装機種）です。配置によっては16ページ、スプライトジェネレータの節約で32ページ、64ページなどの割り付けも可能ですが、BASICではサポートしていません。

この画面モードでは、02020Hから32バイトがパレットテーブルとして使われます（MSXのシステムソフトウェアが設定する）。

4.4 GRAPHIC 1 (SCREEN 1)

4.4.1 特徴

■ 1パターンのサイズ	横 8 ドット　縦 8 ドット
■パターンの数	256 種類
■画面上のパターン数	横 32 パターン　縦 24 パターン
■パターンの色	512 色中 16 色(全画面)
■スプライト	モード 1
■ 1 画面表示に必要な VRAM 容量	4K バイト

4.4.2 関係するレジスタと VRAM の領域

■パターンのフォント	VRAM パターンジェネレータテーブル
■画面上のパターンの位置	VRAM パターンネームテーブル
■パターンの 1 の部分の色コード ■パターンの 0 の部分の色コード	VRAM カラーテーブル 8 パターンごとに 1 組指定可能
■バックドロップの色コード	R＃7　下位 4 ビット
■スプライト	VRAM スプライトアトリビュートテーブル VRAM スプライトパターンテーブル

4.4.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	0	0	0	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	0	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	$\overline{NT}$	DC

□は表示モード設定用のビット（M5〜M1）をGRAPHIC 1モード（00000）にセットした例です。その他のビットは任意に設定します。MSXのシステムソフトウェアはモードレジスタをそれぞれ00000000、01100000、00001000、00001000の値で初期化します。

2. パターンジェネレータテーブルの設定

パターンジェネレータテーブルはパターンのフォントを記憶させるエリアです。

パターンには#0〜#255の番号がつけられ、パターンを画面に表示するときには、このパターン番号で指定します。

パターンジェネレータテーブルの先頭アドレスはレジスタR#4にセットします。指定できるのは、先頭アドレスの上位6ビット（A16〜A11）のみで、下位11ビット（A10〜A0）は「0」とみなされます。したがって、パターンジェネレータテーブルの先頭アドレスとして指定できるのは00000Hから2Kバイト単位の位置になります。

1個のパターンのフォントは8バイトで構成されます。

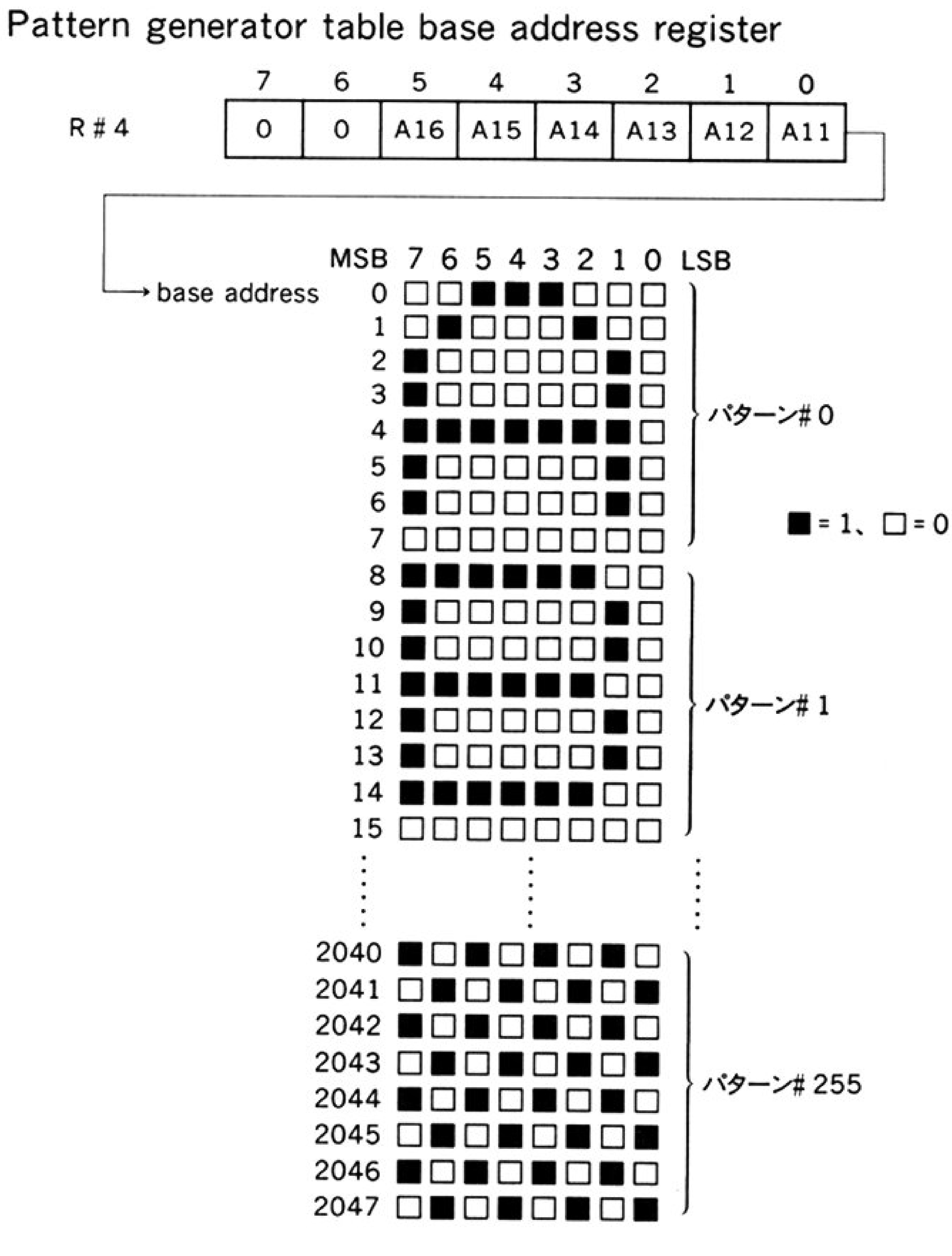


図4.12 GRAPHIC 1モードのパターンジェネレータテーブル例

3. パターンネームテーブルの設定

パターンネームテーブルは 1 バイトが画面上の 1 パターンに対応しています。このテーブルに 0〜255 のパターン番号を書き込むと、対応する画面上の位置に指定したパターンが表示されます。

パターンネームテーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは、先頭アドレスの上位 7 ビット（A16〜A10）のみで、下位 10 ビット（A9〜A0）は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは 00000H から 1K バイト単位の位置になります。

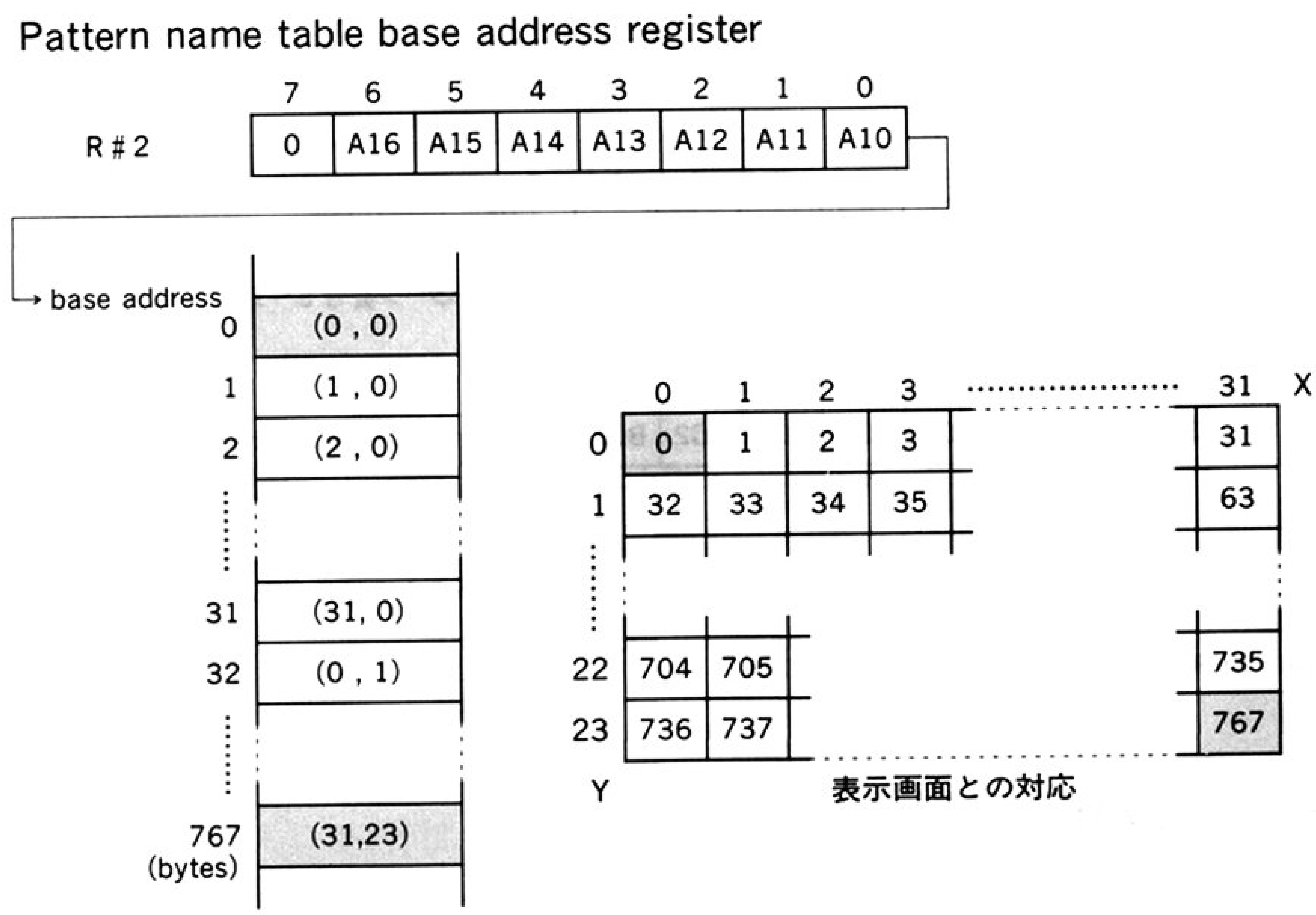


図 4.13　GRAPHIC 1 モードのパターンネームテーブル

4. カラーテーブルの設定

パターンの「1」の部分の色と「0」の部分の色を8パターンごとに1組設定します。

カラーテーブルの先頭アドレスはレジスタR#3とR#10にセットします。カラーテーブルの先頭アドレスとして指定できるのは、00000Hから64バイト単位の位置になります。

Color table base address register

	7	6	5	4	3	2	1	0
R#3	A13	A12	A11	A10	A9	A8	A7	A6
R#10	0	0	0	0	0	A16	A15	A14

→ base address

	7	6	5	4	3	2	1	0	
0	FC3	FC2	FC1	FC0	BC3	BC2	BC1	BC0	パターン番号0〜7
1	FC3	FC2	FC1	FC0	BC3	BC2	BC1	BC0	パターン番号8〜15
⋮									
31	FC3	FC2	FC1	FC0	BC3	BC2	BC1	BC0	パターン番号248〜255

BC3〜BC0: パターンの0の部分の色コード
FC3〜FC0: パターンの1の部分の色コード

図4.14 GRAPHIC 1 モードのカラーテーブル

5. カラーレジスタの設定

R#7で画面のバックドロップの色が指定できます。

Text color / Back drop color register

	7	6	5	4	3	2	1	0
R#7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0

BD3〜BD0: 画面のバックドロップの色を指定
TC3〜TC0: 無効

6. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタR#5とR#11に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタR#6にセットします。詳しくは、「6.1 スプライトモード1」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	A9	A8	A7
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.4.4 GRAPHIC 1(SCREEN 1)モード VRAMマップ

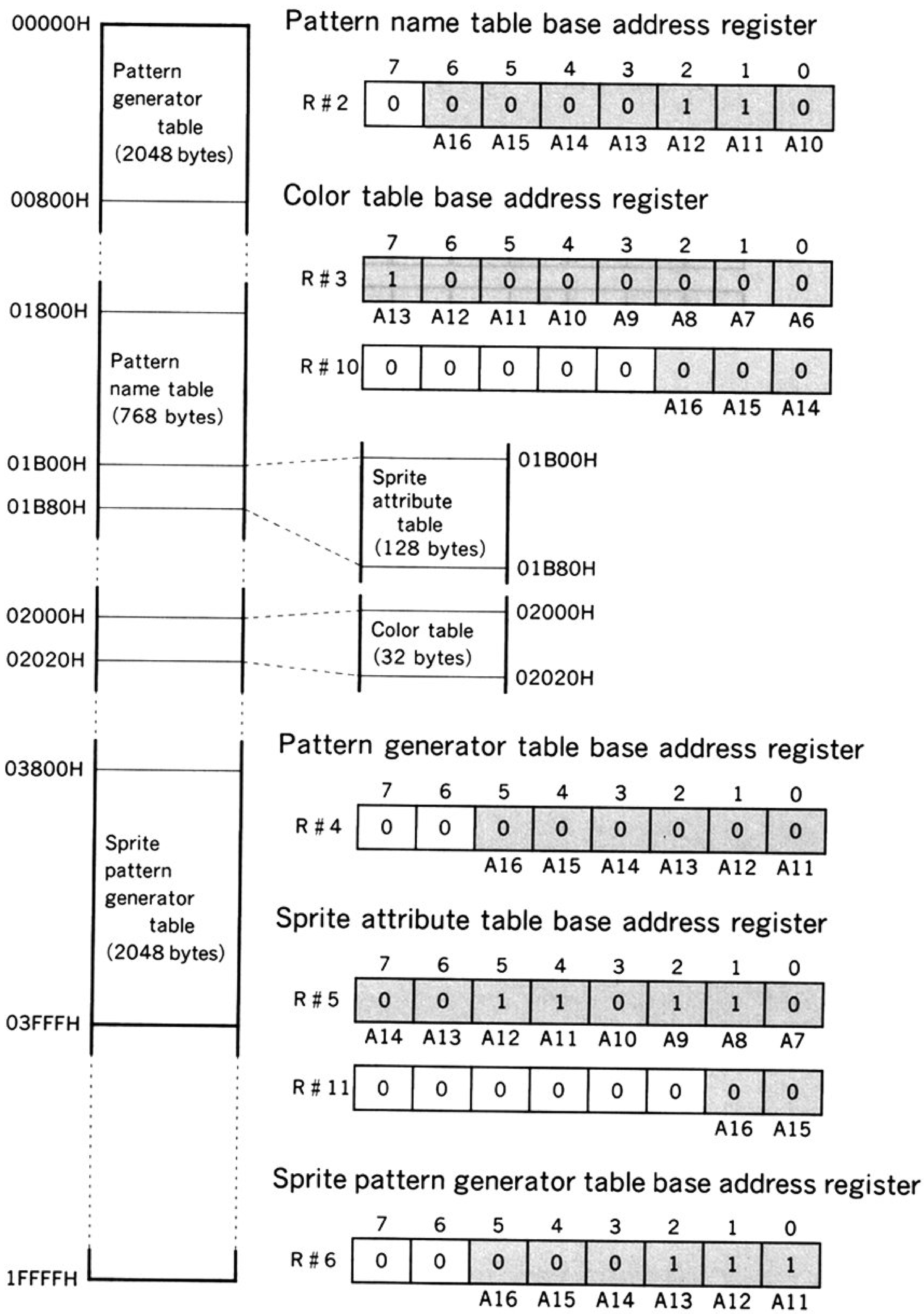


図4.15 GRAPHIC 1モードのVRAMマップ

以下同様に8ページまで割り付け可能(VRAM 128K実装機種)です。配置によっては32ページなどの割り付けも可能ですが、BASICではサポートしていません。

この画面モードでは、02020Hから32バイトがパレットテーブルとして使われます(MSXのシステムソフトウェアが設定する)。

4.5 GRAPHIC 2 (SCREEN 2)、GRAPHIC 3 (SCREEN 4)

4.5.1 特徴

■ 1パターンのサイズ	横 8 ドット　縦 8 ドット
■パターンの数	768 種類
■画面上のパターン数	横 32 パターン　縦 24 パターン
■パターンの色	512 色中 16 色(全画面)
■スプライト	モード 1 (GRAPHIC 2) モード 2 (GRAPHIC 3)
■ 1 画面表示に必要な VRAM 容量	16K バイト

GRAPHIC 2 と GRAPHIC 3 の違いは、スプライトのモードだけです。

4.5.2 関係するレジスタと VRAM の領域

■パターンのフォント	VRAM パターンジェネレータテーブル
■画面上のパターンの位置	VRAM パターンネームテーブル
■パターンの 1 の部分の色コード ■パターンの 0 の部分の色コード	VRAM カラーテーブル 各パターンの 1 ラスタに 1 組指定可能
■バックドロップの色コード	R#7　下位 4 ビット
■スプライト	VRAM スプライトアトリビュートテーブル VRAM スプライトパターンテーブル

4.5.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	0	*	*	0

(bits 2–1) 01 GRAPHIC 2
10 GRAPHIC 3

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	0	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	E0	$\overline{NT}$	DC

▨は表示モード設定用のビット(M5～M1)をGRAPHIC 2(00100)またはGRAPHIC 3(01000)モードにセットした例です。その他のビットは任意に設定します。MSXのシステムソフトウェアはモードレジスタをそれぞれ00000010(GRAPHIC 3のときは00000100)、01100000、00001000、00001000の値で初期化します。

2. パターンジェネレータテーブルの設定

パターンジェネレータテーブルはパターンを記憶させるエリアです。

パターンには#0～#255の番号（パターンネーム）がつけられていますが、画面を3分割して、それぞれのブロックごとに1つのパターンジェネレータテーブルを持つことにより、768種類のパターンを使用できます。これにより、画面に任意のパターンを表示することができるフルビットマップ表示の機能を実現しています。

1個のパターンは8バイトで構成されるので、合計6136バイトの領域を占めることになります。

パターンジェネレータテーブルの先頭アドレスはレジスタR#4にセットします。指定できるのは、先頭アドレスの上位4ビット（A16～A13）のみで、下位13ビット（A12～A0）は「0」とみなされます。したがって、パターンジェネレータテーブルの先頭アドレスとして指定できるのは00000Hから8Kバイト単位の位置になります。

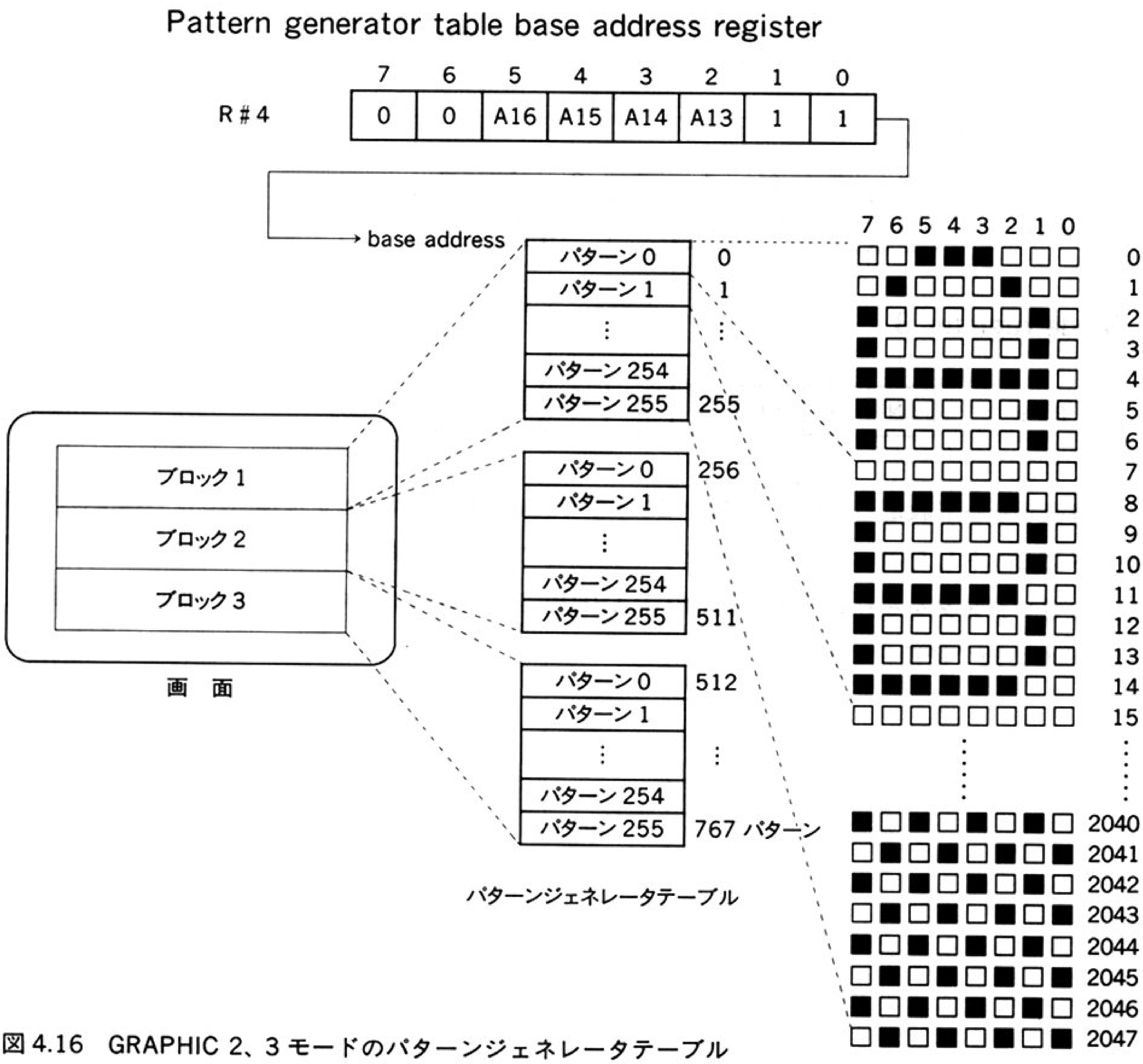


図4.16 GRAPHIC 2、3モードのパターンジェネレータテーブル

3. カラーテーブルの設定

カラーテーブルはパターンジェネレータテーブルと1対1に対応し、パターンの「1」の部分の色と「0」の部分の色を各パターンの1ラスタに対して1組指定します。したがって、1つのパターンに対して8バイトのデータとなり、合計でパターンジェネレータテーブルと同じ6136バイトの領域を占めます。

カラーテーブルの先頭アドレスはレジスタR#3とR#10にセットします。カラーテーブルの先頭アドレスとして指定できるのは、00000Hから8Kバイトの位置になります。

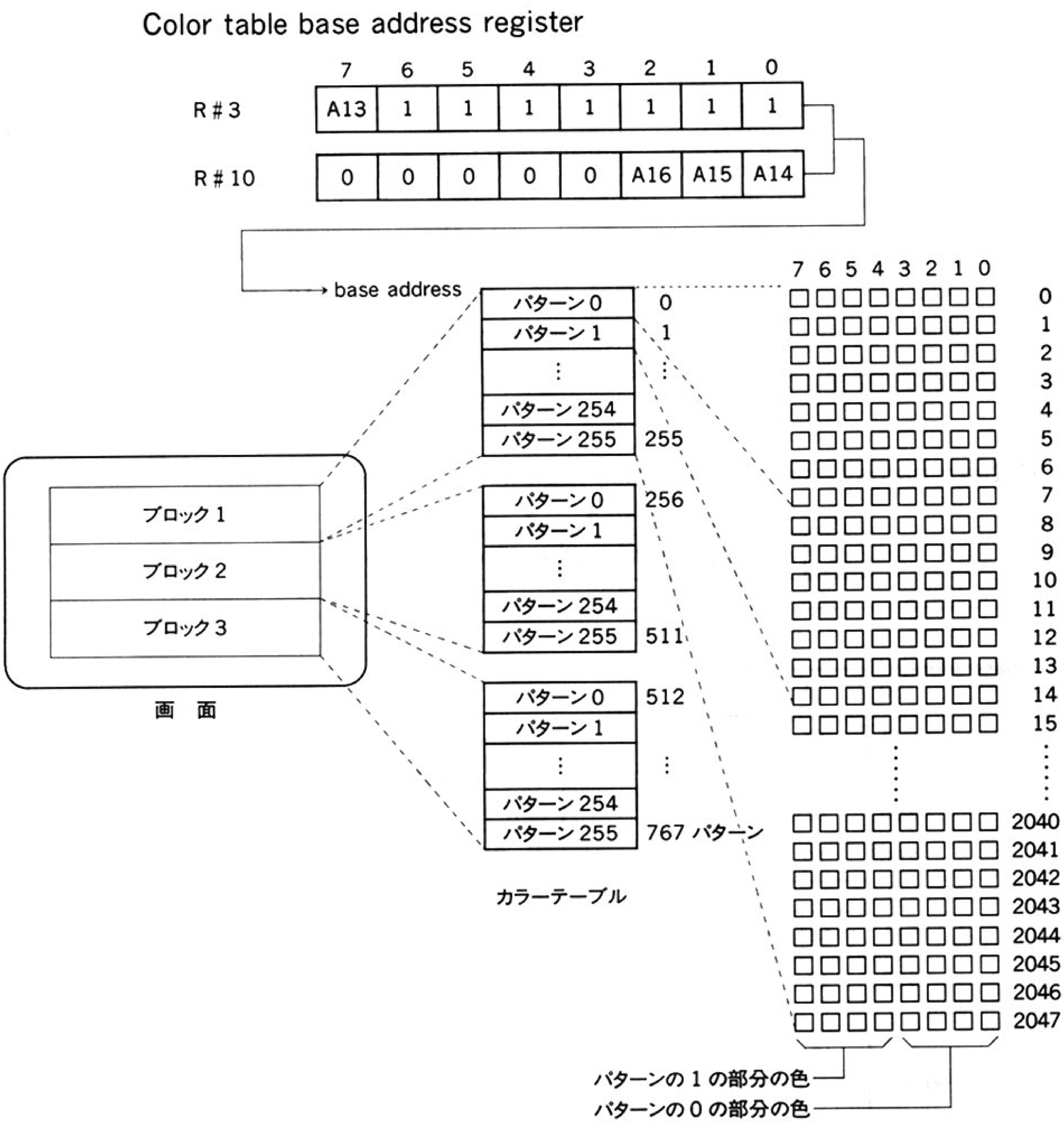
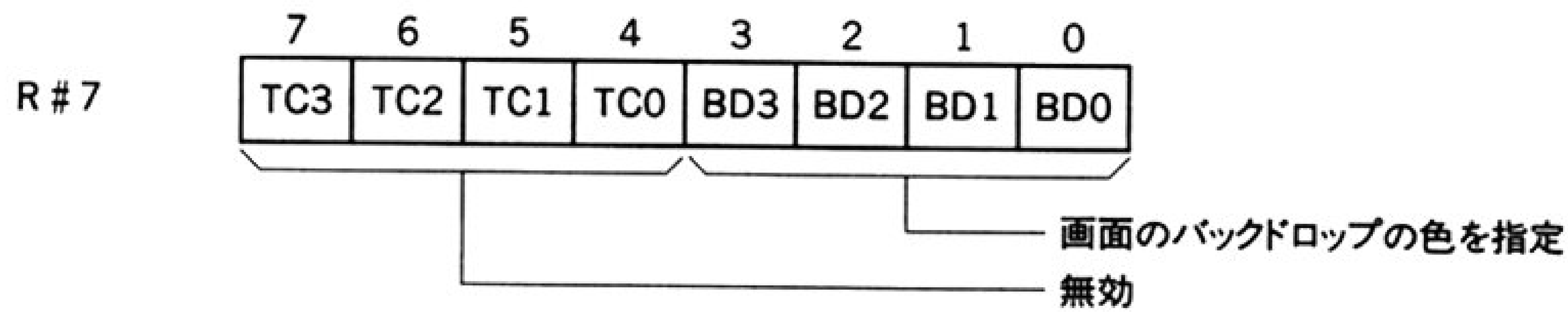


図4.17 GRAPHIC 2、3モードのカラーテーブル

4. カラーレジスタの設定

R#7 で画面のバックドロップの色が指定できます。

Text color / Back drop color register



5. パターンネームテーブルの設定

パターンネームテーブルは 1 バイトが画面上の 1 パターンに対応しています。このテーブルに 0～255 のパターン番号を書き込むと、対応する画面上の位置に指定したパターンが表示され、画面を上、中、下と 3 分割して使うことにより、768 種類のパターンを表示することができます。

パターンネームテーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは、先頭アドレスの上位 7 ビット（A16～A10）のみで、下位 10 ビット（A9～A0）は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは 00000H から 1K バイト単位の位置になります。

Pattern name table base address register

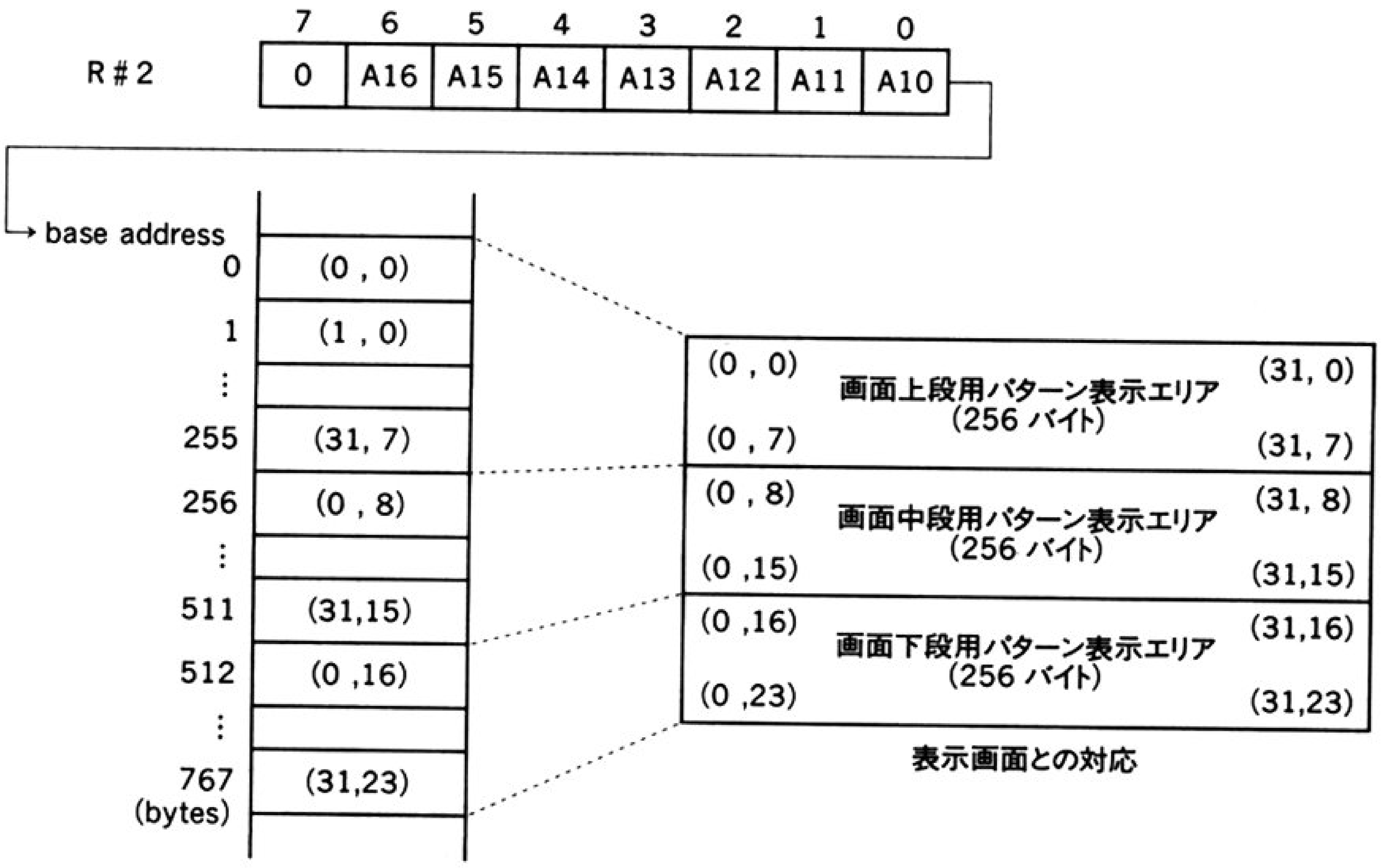


図 4.18 GRAPHIC 2、3 のパターンネームテーブル

6. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタ R#5 と R#11 に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタ R#6 にセットします。GRAPHIC 2 モードのときは「6.1 スプライトモード 1」の項を、GRAPHIC 3 モードのときは「6.2 スプライトモード 2」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	A9	A8	A7
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.5.4 GRAPHIC 2 (SCREEN 2)モードの VRAM マップ

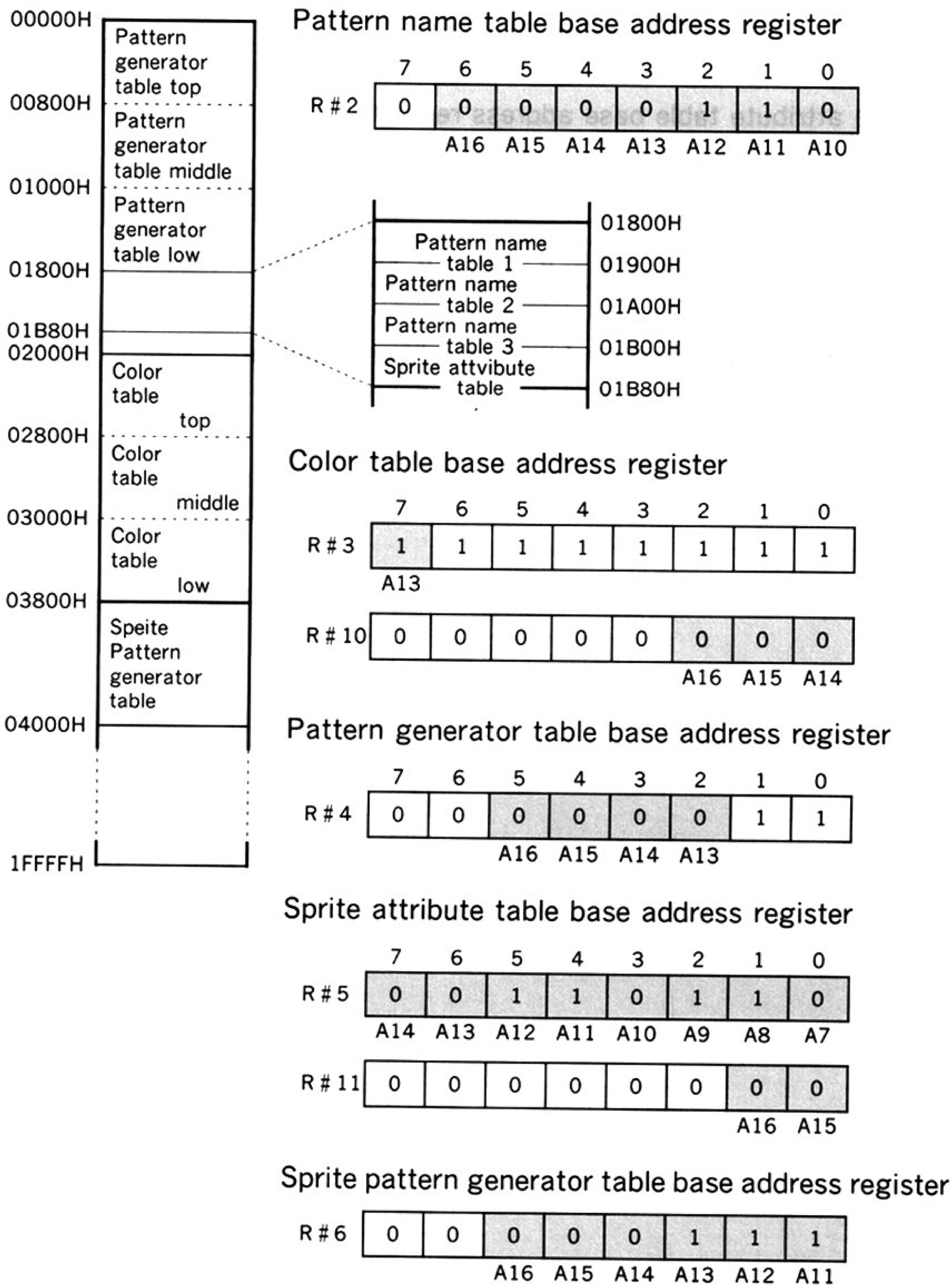


図4.19 GRAPHIC 2 の VRAM マップ

以下同様に、最高 8 ページまで割り付け可能（VRAM 128K 実装機種）です。しかし、BASIC ではサポートしていません。

この画面モードでは、01B80H から 32 バイトがパレットテーブルとして使われます（MSX のシステムソフトウェアが設定する）。

表示エリアが 256×192 のとき図 4.19 のとおりですが、ハードウェアスクロールの利用によって表示エリアを 256×256 とする場合には、パターンネームテーブルは 768 バイトではなく 1024 バイト連続しており、1024 種類のパターンを用意できることになります。また、この場合はパターンジェネレータテーブルとカラーテーブルもそれぞれ連続した 8K バイトとなり、スプライトジェネレータテーブル、パターンネームテーブルなどは別のページに移動しなければなりません。

4.5.5 GRAPHIC 3 (SCREEN 4)モードのVRAMマップ

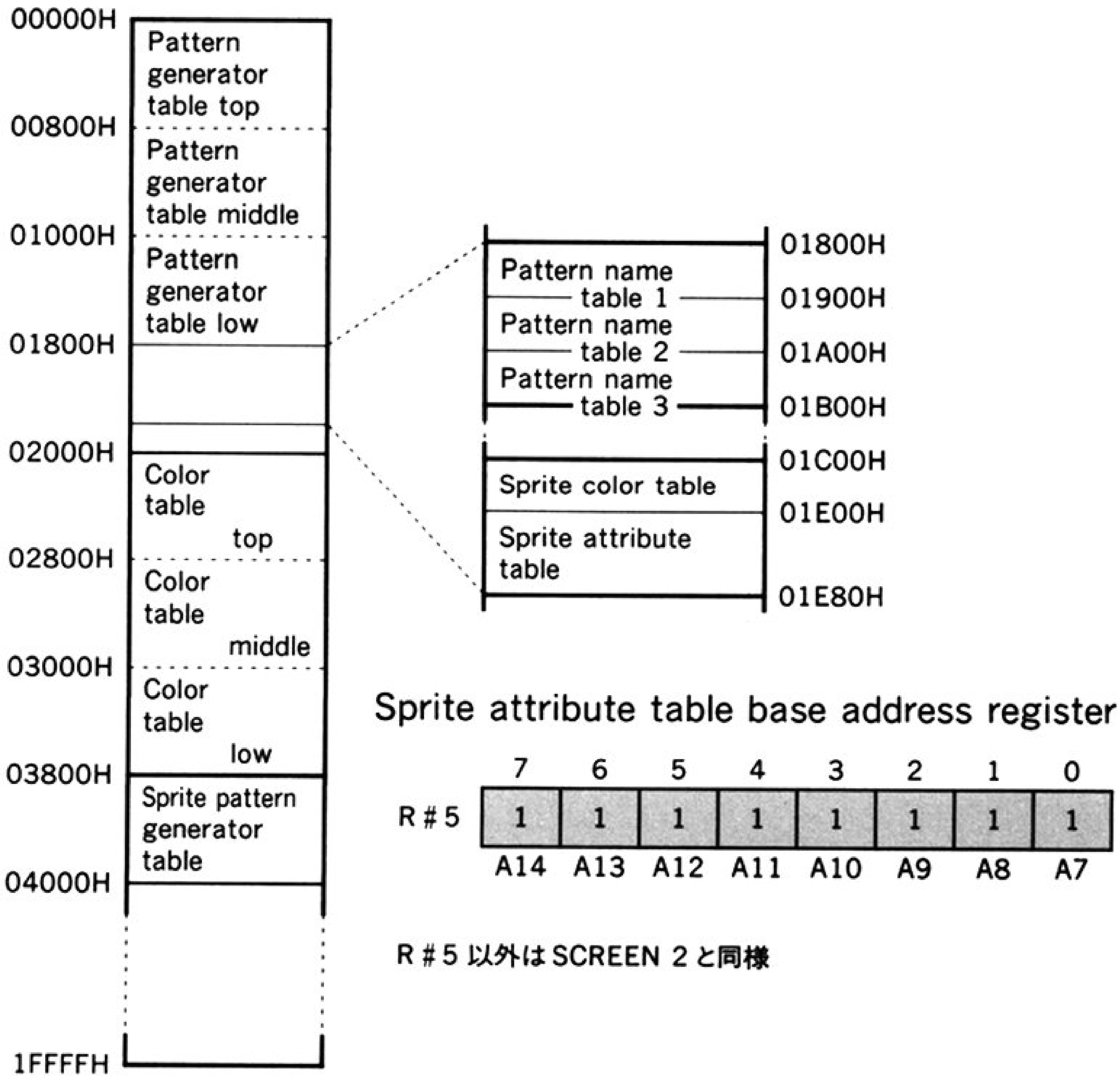


図4.20 GRAPHIC 3のVRAMマップ

以下同様に、最高8ページまで割り付けが可能(VRAM 128K実装機種)です。しかし、BASICではサポートしていません。

この画面モードでは、01B80Hから32バイトがパレットテーブルとして使われます(MSXのシステムソフトウェアが設定する)。

4.6 GRAPHIC 4 (SCREEN 5)

4.6.1 特徴

■解像度	横 256 ドット×縦 212 ドット(または縦 192 ドット)のビットマップグラフィックモード
■表示色	512 色中 16 色 (全画面)
■スプライト	モード 2
■ 1 画面表示に必要な VRAM 容量	32K バイト

4.6.2 関係するレジスタと VRAM の領域

■グラフィックス	VRAM	パターンネームテーブル
■バックドロップの色コード	R#7	下位 4 ビット
■スプライト	VRAM	スプライトアトリビュートテーブル
	VRAM	スプライトパターンテーブル

4.6.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	0	1	1	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	0	0	0	SI	MAG

Mode register 2

R#8	7	6	5	4	3	2	1	0
	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

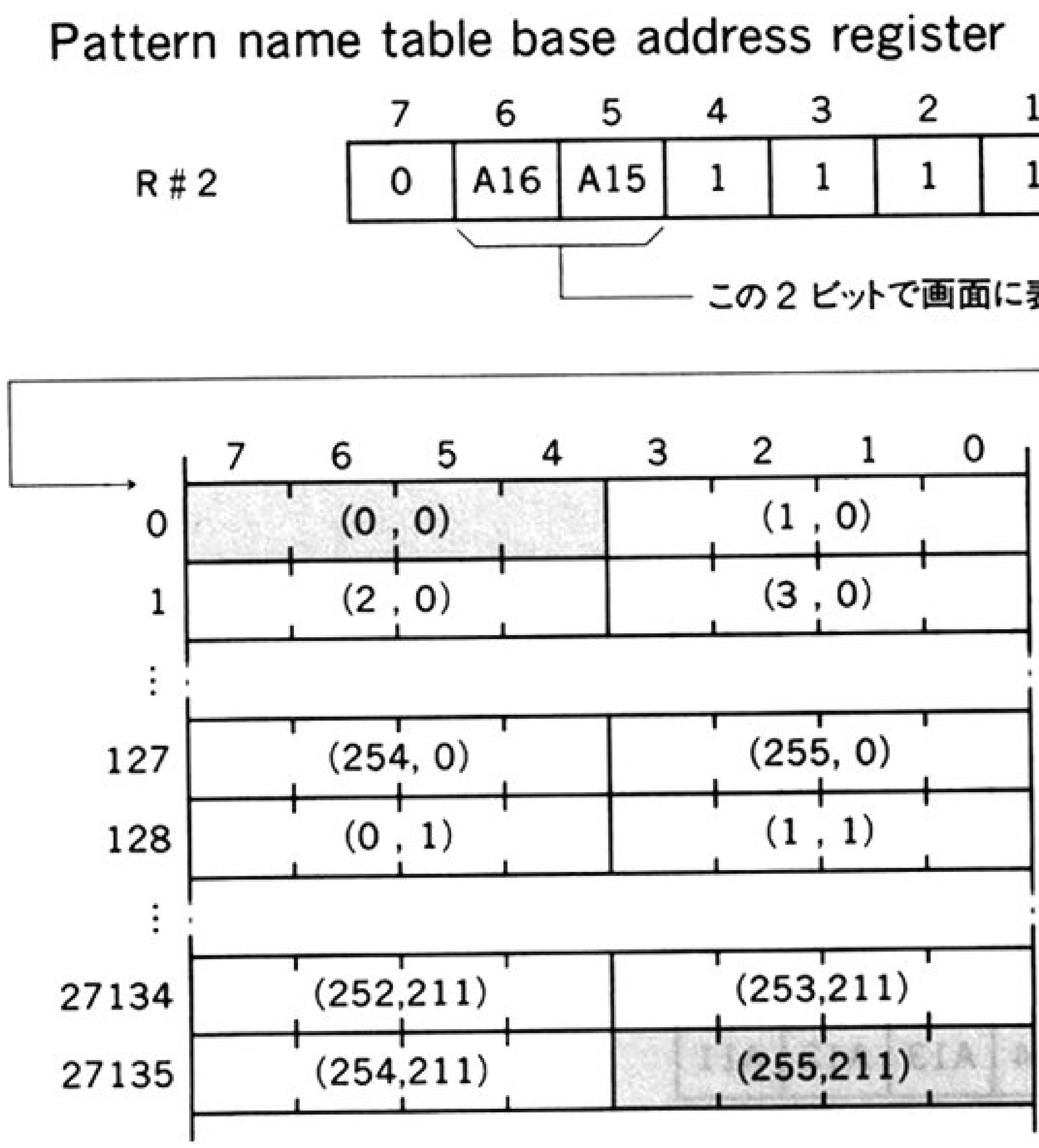
R#9	7	6	5	4	3	2	1	0
	LN	0	S1	S0	IL	EO	$\overline{NT}$	DC

▢は表示モード設定用のビット（M5〜M1）をGRAPHIC 4（01100）モードにセットした例です。この表示モードでは、LN＝1のとき縦212ドット、LN＝0のとき縦192ドットに設定されます。その他のビットは任意に設定します。MSXのシステムソフトウェアはモードレジスタをそれぞれ00000110、01100000、00001000、10001000の値で初期化します。

2. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の2ドットに対応しており、1ドットごとにパレットにより選択された512色中の16色を選んで表示することができます。

パターンネームテーブルの先頭アドレスはレジスタR#2にセットします。指定できるのは、上位2ビット（A16〜A15）のみで、下位15ビットは「0」とみなされます。したがって、パターンネームテーブルが設定可能なアドレスは、00000H、08000H、10000H、18000Hの4箇所になります。



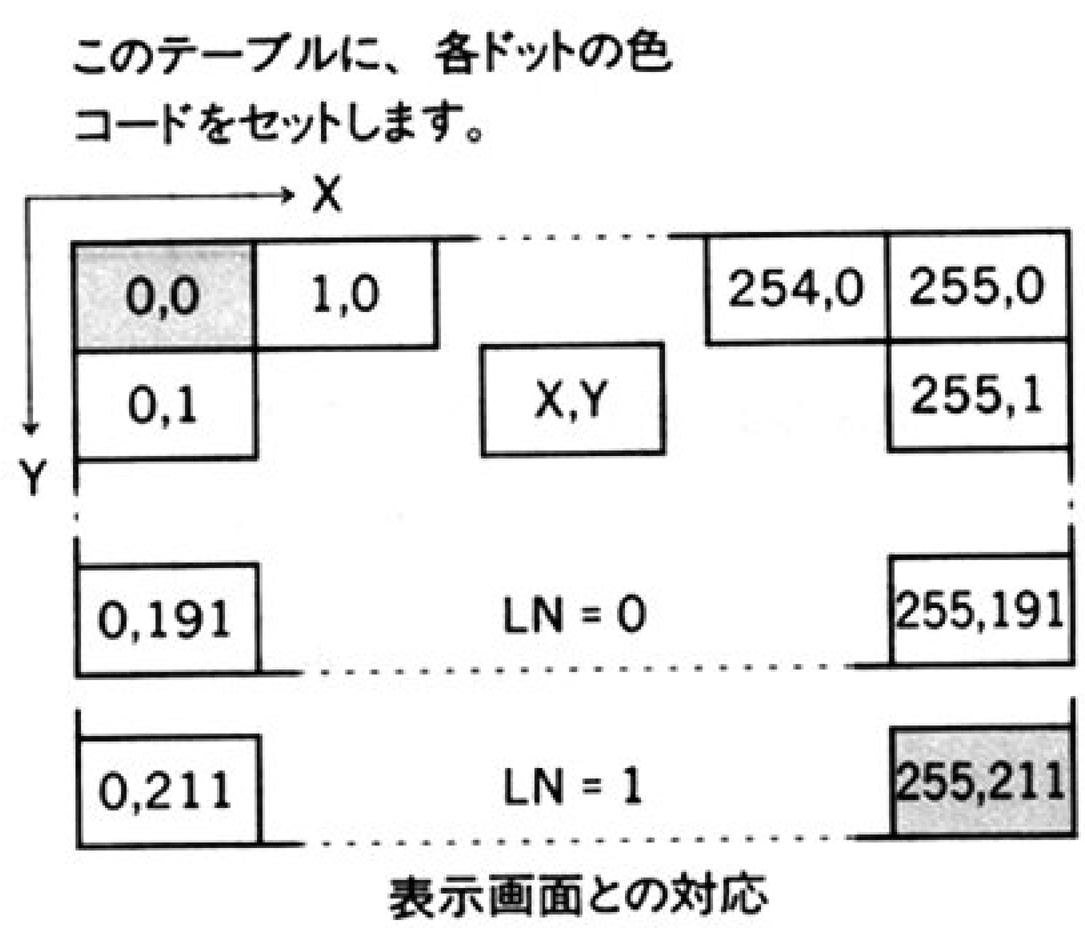


図4.21 GRAPHIC 4 モードのパターンネームテーブル

3. カラーレジスタの設定

R#7で画面のバックドロップの色が指定できます。

Text color / Back drop color register

	7	6	5	4	3	2	1	0
R#7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0

- TC3〜TC0：無効
- BD3〜BD0：画面のバックドロップの色を指定

4. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタR#5とR#11に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタR#6にセットします。詳しくは、「6.2 スプライトモード2」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	1	1	1
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.6.4 GRAPHIC 4 (SCREEN 5)モードの VRAM マップ

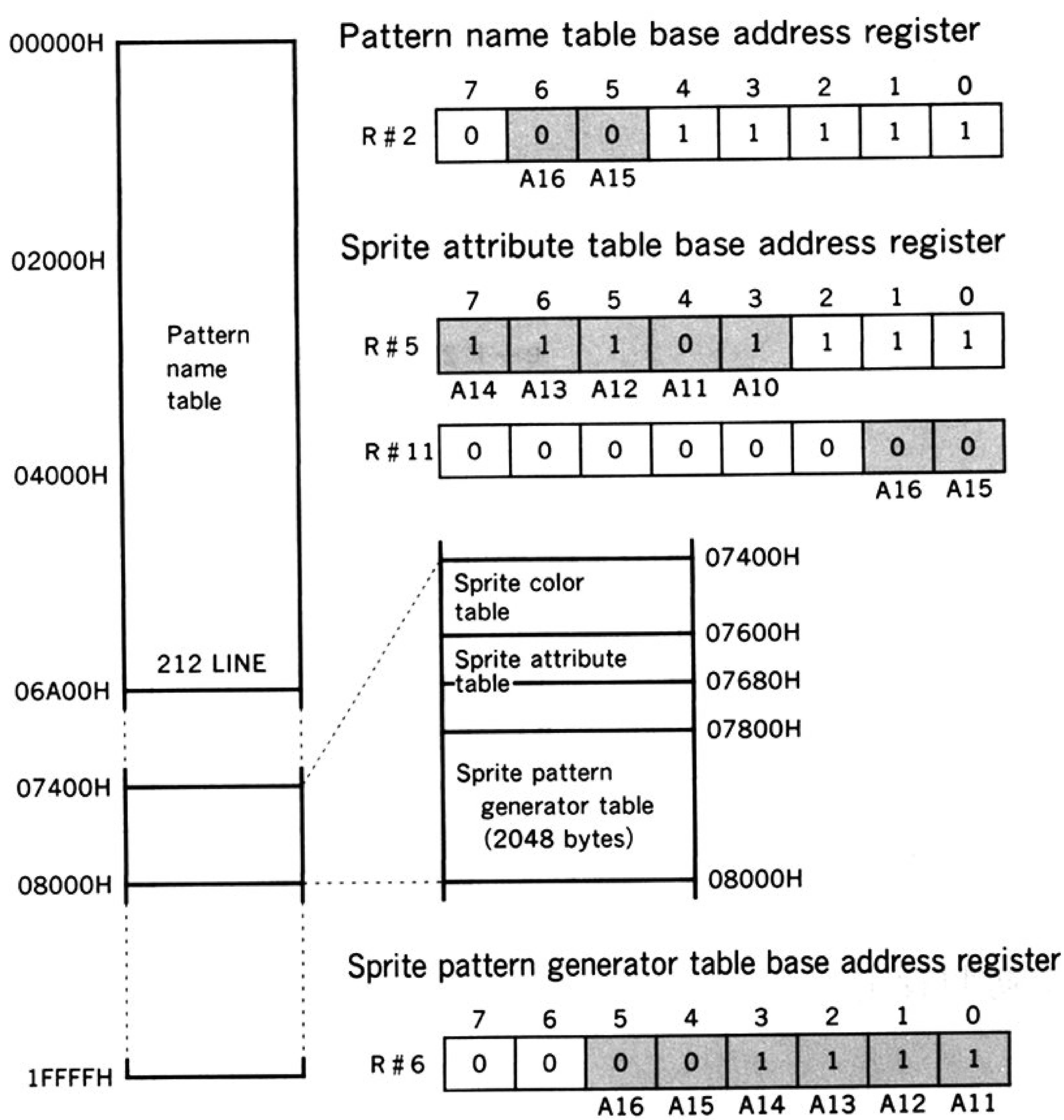


図 4.22 GRAPHIC 4 モードの VRAM マップ

以下同様に、4 ページまで割り付け可能（VRAM　128K 実装機種）です。BASIC では SET PAGE 命令で切り換えて表示することができます。

この画面モードでは、07680H から 32 バイトがパレットテーブルとして使われます（MSX のシステムソフトウェアが設定する）。

4.7 GRAPHIC 5 (SCREEN 6)

4.7.1 特徴

■解像度	横512ドット×縦212ドット(または縦192ドット)のビットマップグラフィックモード
■表示色	512色中4色 (全画面)
■スプライト	モード2
■1画面表示に必要なVRAM容量	32Kバイト

4.7.2 関係するレジスタとVRAMの領域

■グラフィックス	VRAM	パターンネームテーブル
■バックドロップの色コード	R#7	下位4ビット
■スプライト	VRAM	スプライトアトリビュートテーブル
	VRAM	スプライトパターンテーブル

4.7.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	1	0	0	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	0	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	$\overline{NT}$	DC

▨は表示モード設定用のビット（M5～M1）をGRAPHIC 5（10000）モードにセットした例です。この表示モードでは、LN＝1のとき縦212ドット、LN＝0のとき縦192ドットに設定されます。その他のビットは任意に設定します。MSXのシステムソフトウェアはモードレジスタをそれぞれ00001000、01100000、00001000、10001000の値で初期化します。

2. パターンネームテーブルの設定

パターンネームテーブルは 1 バイトが画面上の 4 ドットに対応しており、1 ドットごとにパレットにより選択された 512 色中の 4 色から選んで表示することができます。

パターンネームテーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは、先頭アドレスの上位 2 ビット（A16～A15）のみで、下位 15 ビット（A14～A0）は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは 00000H、08000H、10000H、18000H の 4 箇所になります。

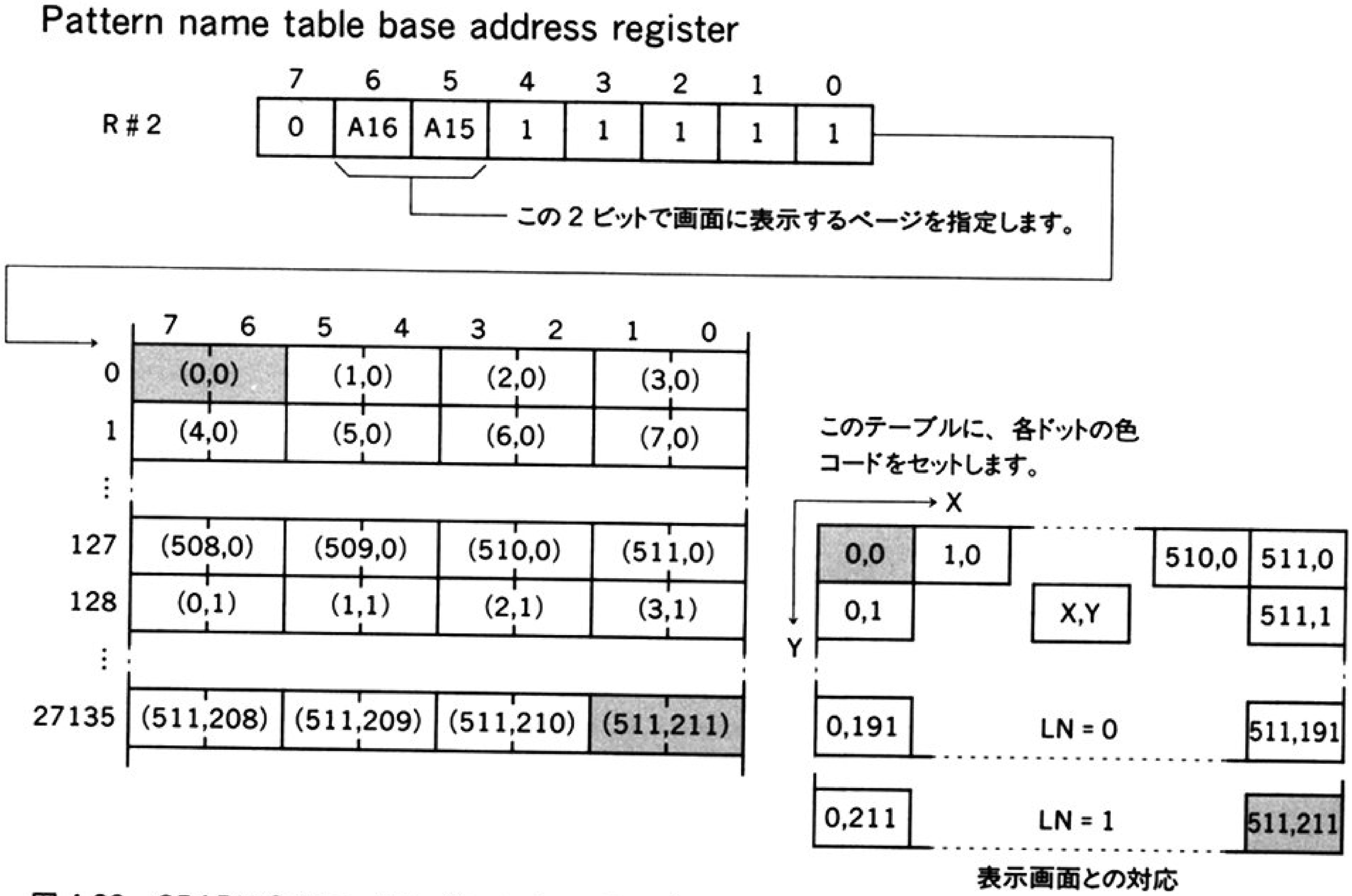
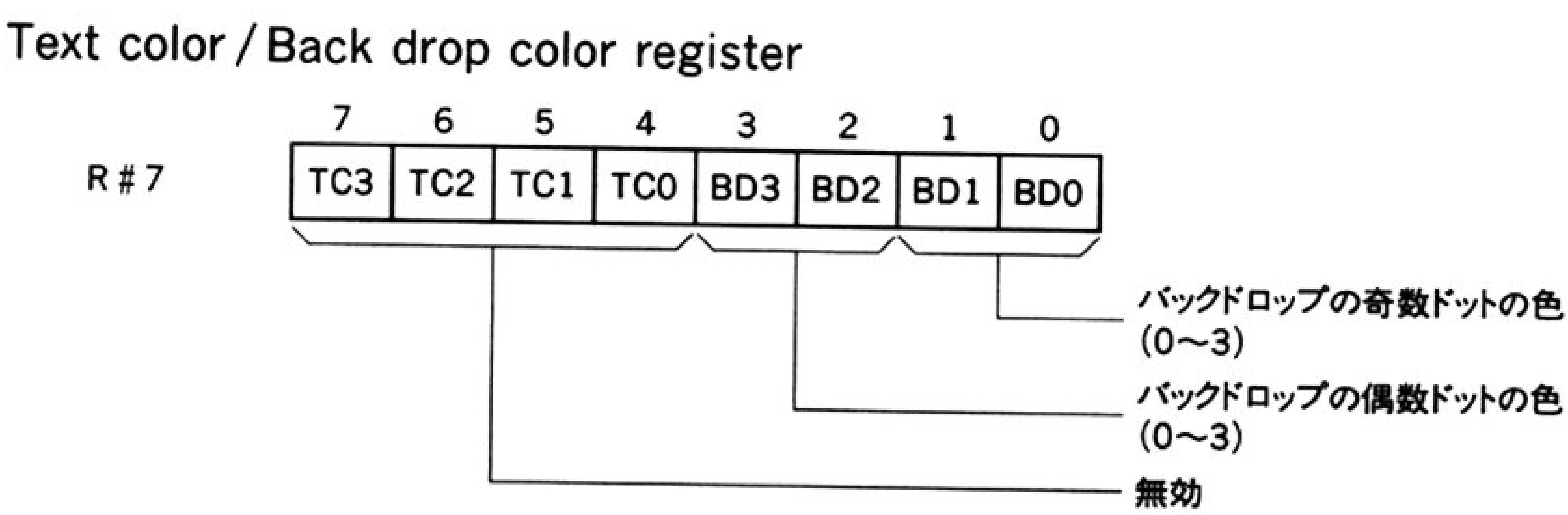


図 4.23　GRAPHIC 5 モードのパターンネームテーブル

3. カラーレジスタの設定

R#7 で画面のバックドロップの色が指定できます。



4. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタR#5とR#11に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタR#6にセットします。詳しくは、「6.2 スプライトモード2」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	1	1	1
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

5. ハードウェアタイリング機能

GRAPHIC 5 モードでは、スプライトとバックドロップの色についてハードウェアタイリングの機能が働きます。これらの色はパターンネームテーブルの場合と異なり 4 ビットで設定します。この 4 ビットの上位 2 ビットを X 座標（0～511）の偶数ドットの色コードに、下位 2 ビットを奇数ドットの色コードに設定します。

GRAPHIC 5 モードでは、スプライトの 1 ドットの大きさはグラフィックのドットの大きさの 2 倍ですが、このタイリング機能によってスプライトの 1 ドットの左半分と右半分にそれぞれ異なる色を表示することができます。

バックドロップの色も同様に偶数ドットと奇数ドットに対して異なる色を指定できます。

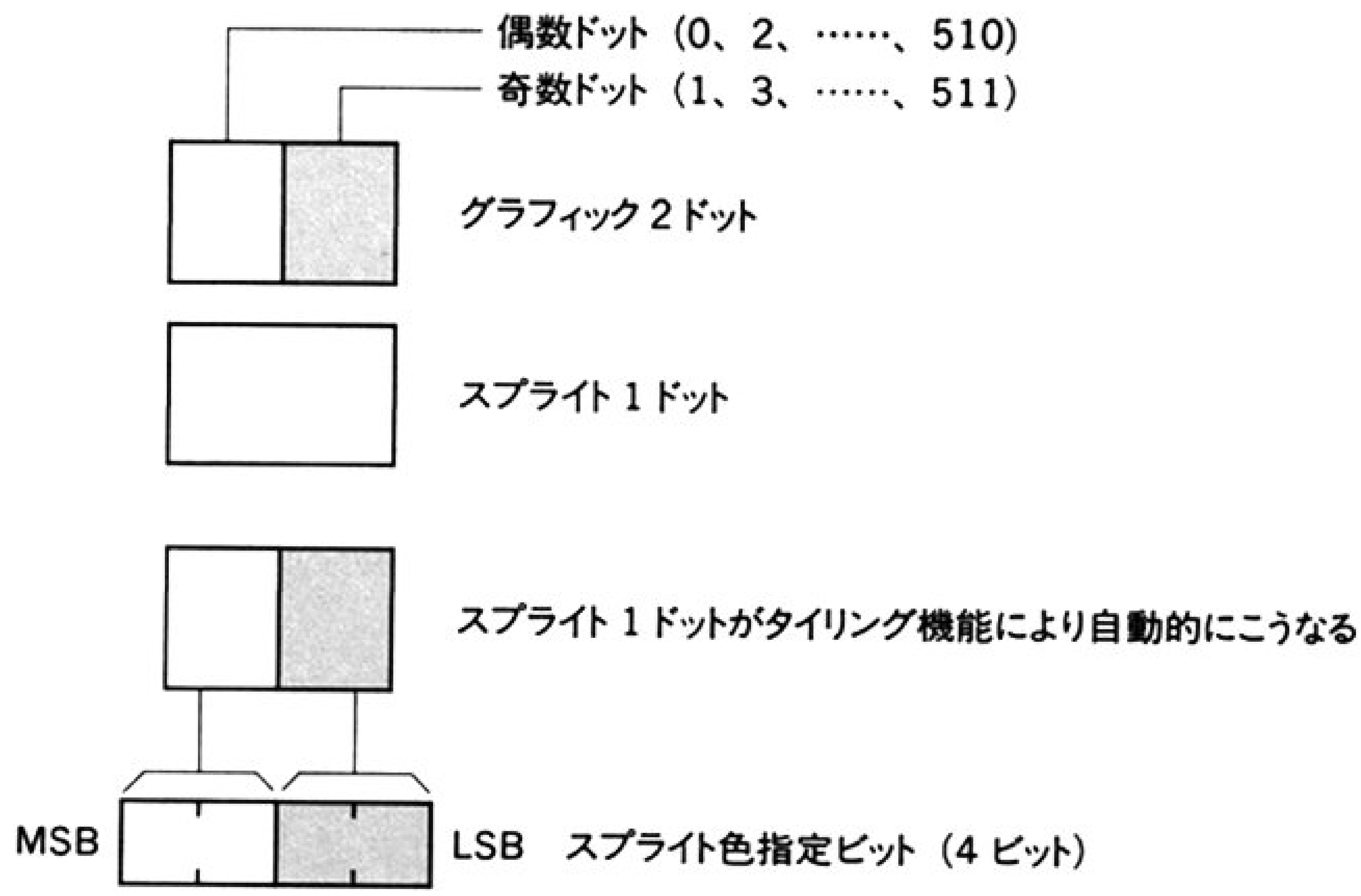


図 4.24 ハードウェアタイリングの設定

4.7.4 GRAPHIC 5 (SCREEN 6)モードの VRAM マップ

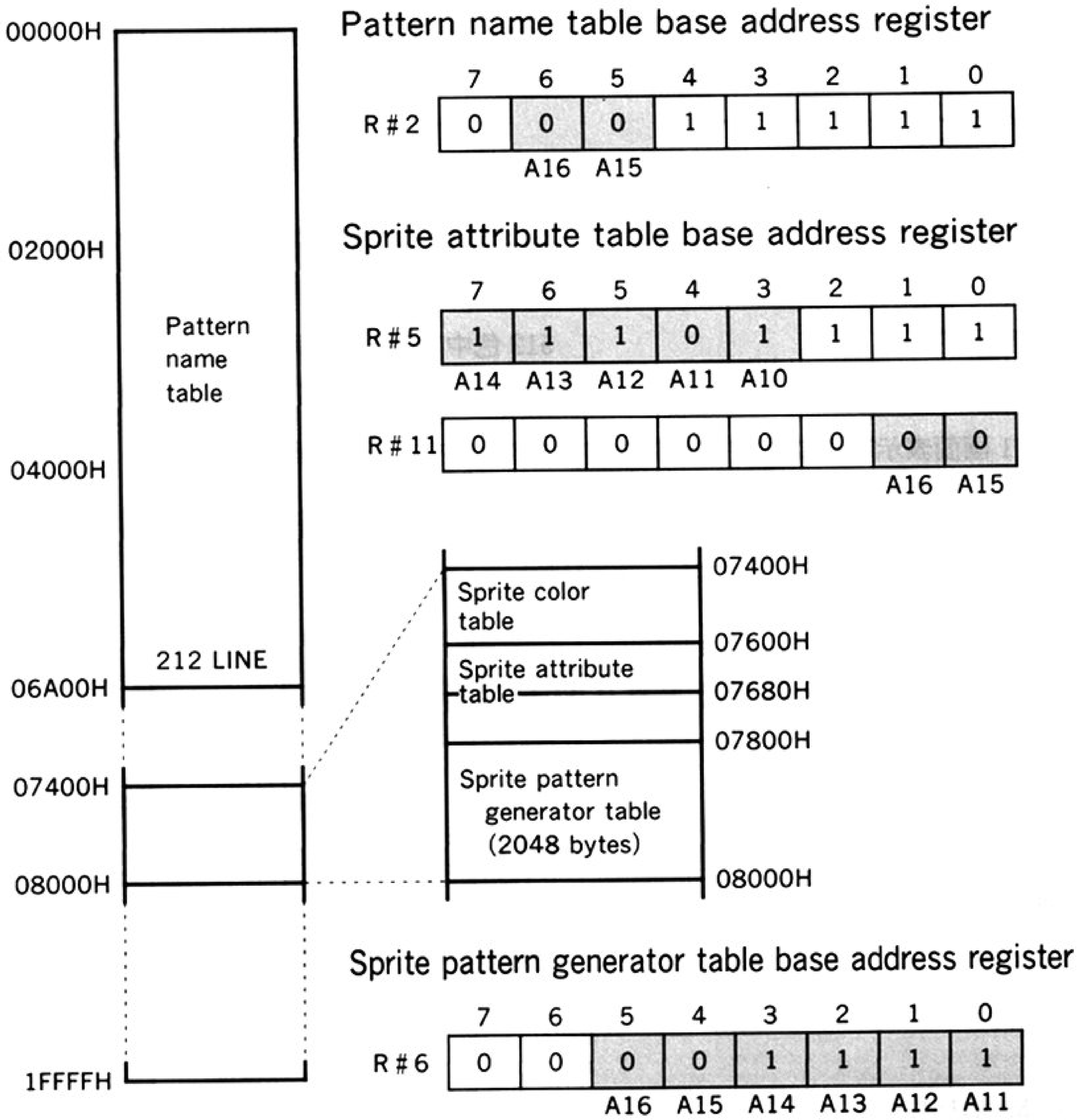


図4.25 GRAPHIC 4 モードの VRAM マップ

以下同様に、4ページまで割り付け可能（VRAM　128K 実装機種）です。BASIC では SET PAGE 命令で切り換えて表示することができます。

この画面モードでは、07680H から 32 バイトがパレットテーブルとして使われます（MSX のシステムソフトウェアが設定する）。

4.8 GRAPHIC 6 (SCREEN 7)

4.8.1 特徴

■解像度	横512ドット×縦212ドット(または縦192ドット)のビットマップグラフィックモード
■表示色	512色中16色 (全画面)
■スプライト	モード2
■1画面表示に必要なVRAM容量	128Kバイト (2画面)

このモードを使用するためには、VRAMが128Kバイト必要です。

4.8.2 関係するレジスタとVRAMの領域

■グラフィックス	VRAM	パターンネームテーブル
■バックドロップの色コード	R#7	下位4ビット
■スプライト	VRAM	スプライトアトリビュートテーブル
	VRAM	スプライト　パターンテーブル

4.8.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	1	0	1	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	0	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	$\overline{NT}$	DC

▨は表示モード設定用のビット（M5～M1）をGRAPHIC 6（10100）モードにセットした例です。この表示モードでは、LN＝1のとき縦212ドット、LN＝0のとき縦192ドットに設定されます。その他のビットは任意に設定します。MSXのシステムソフトウェアはモードレジスタをそれぞれ0001010、01100000、00001000、10001000の値で初期化します。

2. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の2ドットに対応しており、1ドットごとにパレットにより選択された512色中の16色から選んで表示することができます。

パターンネームテーブルの先頭アドレスはレジスタR#2にセットします。指定できるのは、先頭アドレスの上位1ビット（A16）のみで、下位16ビット（A15〜A0）は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは00000Hと10000Hの位置になります。

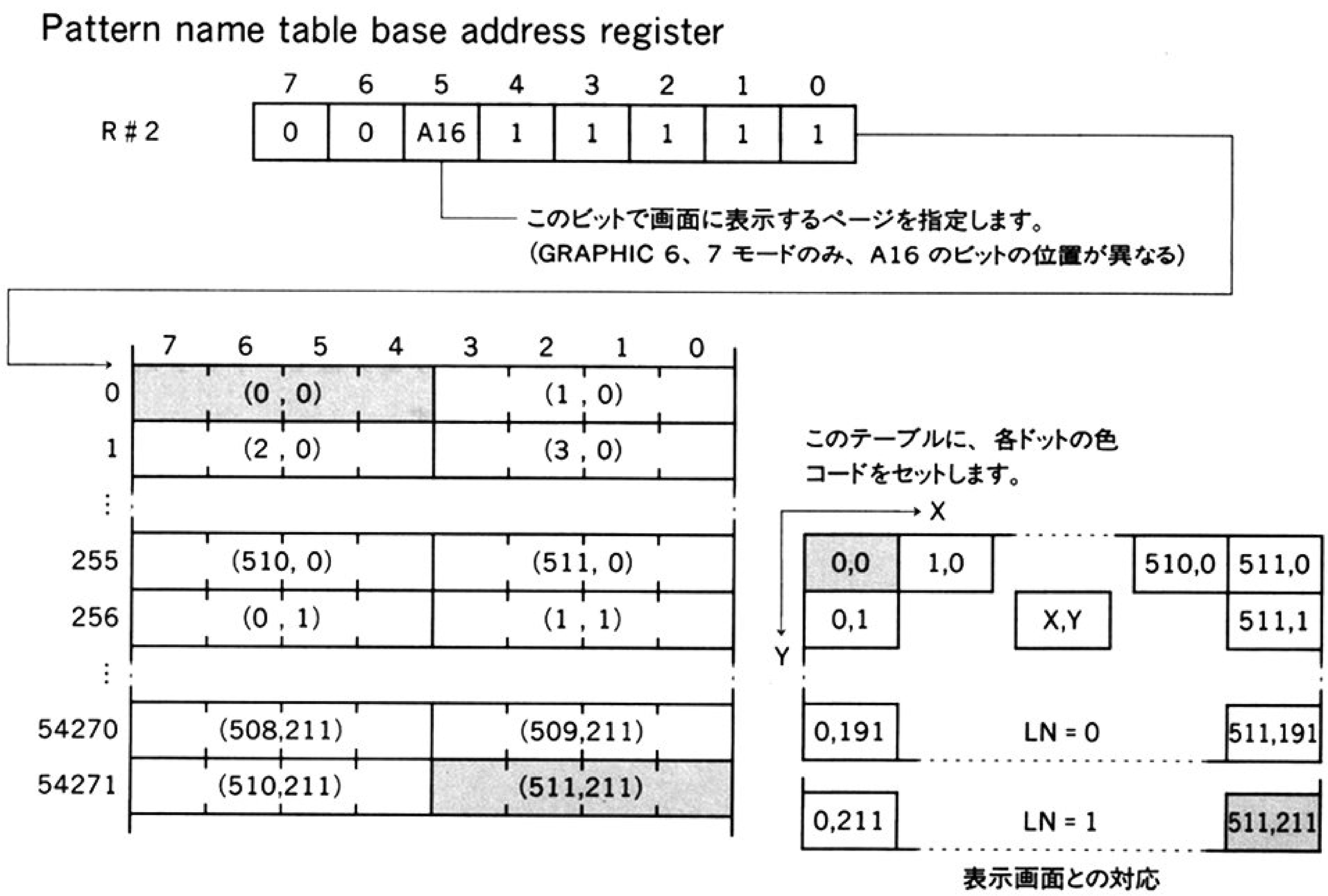
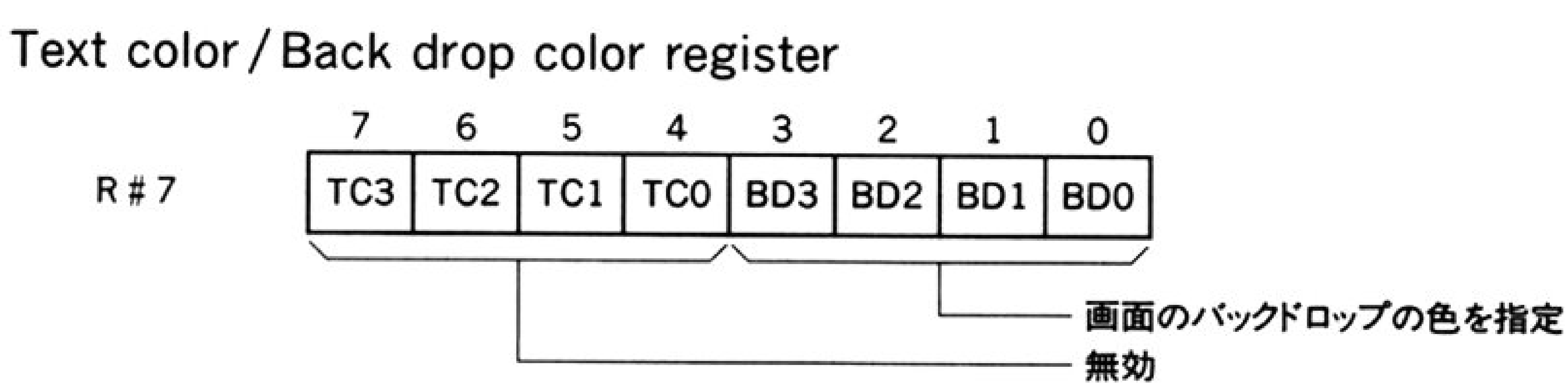


図4.26 GRAPHIC 6 モードのパターンネームテーブル

3. カラーレジスタの設定

R#7で画面のバックドロップの色が指定できます。



4. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタ R#5 と R#11 に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタ R#6 にセットします。詳しくは、「6.2 スプライトモード 2」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	1	1	1
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.8.4 GRAPHIC 6 (SCREEN 7)モードのVRAMマップ

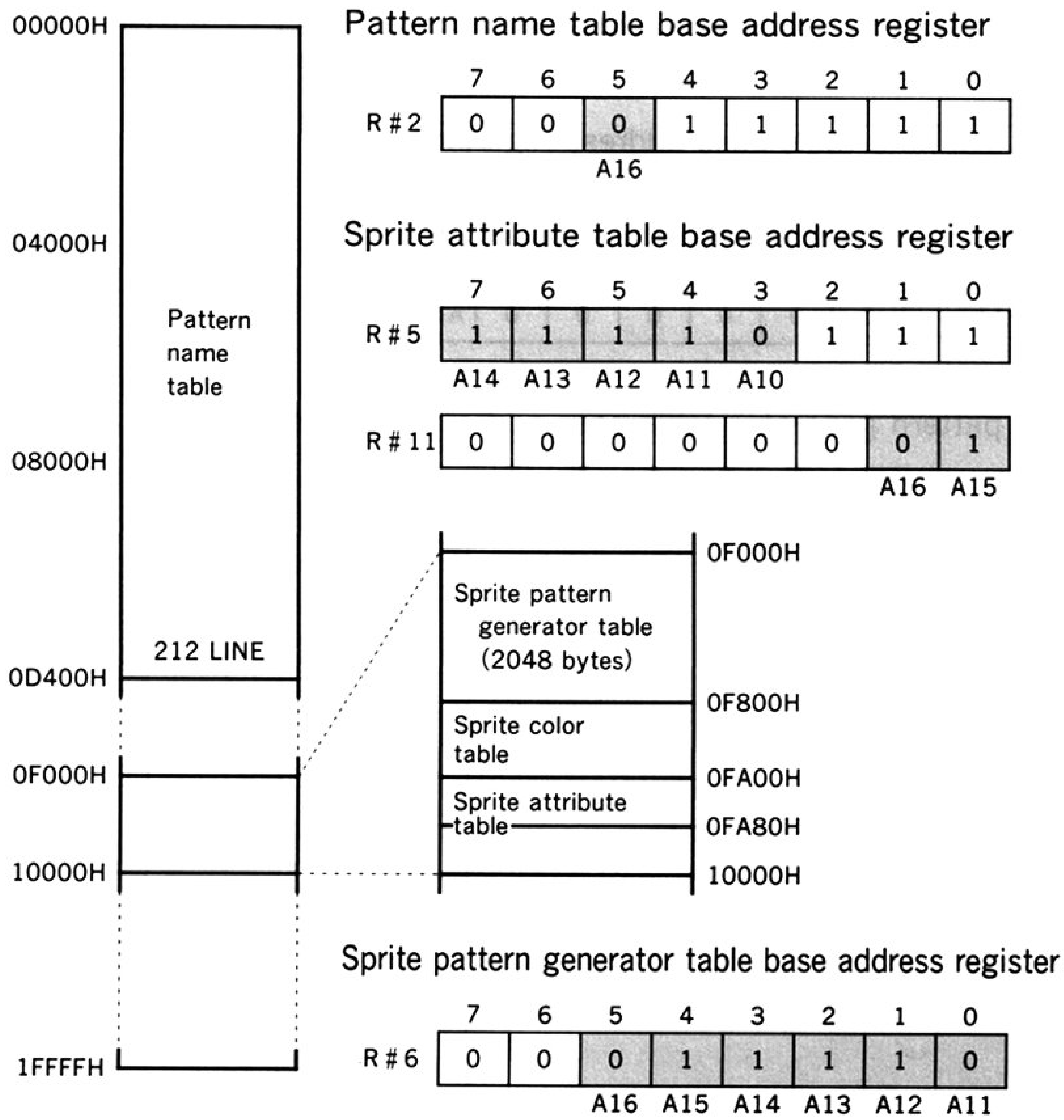


図4.27 GRAPHIC 6モードのVRAMマップ

以下同様に、2ページまで割り付け可能（VRAM 128K実装機種）です。BASICではSET PAGE命令で切り換えて表示することができます。

この画面モードでは、0FA80Hから32バイトがパレットテーブルとして使われます(MSXのシステムソフトウェアが設定する)。

4.9 GRAPHIC 7 (SCREEN 8)

4.9.1 特徴

■解像度	横 256 ドット×縦 212 ドット(または縦 192 ドット)のビットマップグラフィックモード
■表示色	256 色同時 (全画面)
■スプライト	モード 2
■ 1 画面表示に必要な VRAM 容量	128K バイト (2 画面)

このモードを使用するためには、VRAM が 128K バイト必要です。

4.9.2 関係するレジスタと VRAM の領域

■グラフィックス	VRAM	パターンネームテーブル
■バックドロップの色コード	R#7	下位 4 ビット
■スプライト	VRAM	スプライトアトリビュートテーブル
	VRAM	スプライトパターンテーブル

4.9.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	1	1	1	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	0	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	$\overline{NT}$	DC

□は表示モード設定用のビット（M5～M1）をGRAPHIC 7（11100）モードにセットした例です。この表示モードでは、LN＝1のとき縦212ドット、LN＝0のとき縦192ドットに設定されます。その他のビットは任意に設定します。MSXのシステムソフトウェアはモードレジスタをそれぞれ00001110、01100000、00001000、10001000の値で初期化します。

2. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の1ドットに対応しており、1ドットごとに256色表示することができます。

パターンネームテーブルの先頭アドレスはレジスタR#2にセットします。指定できるのは、先頭アドレスの上位1ビット(A16)のみで、下位16ビット(A15〜A0)は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは00000Hと10000Hの位置になります。

Pattern name table base address register

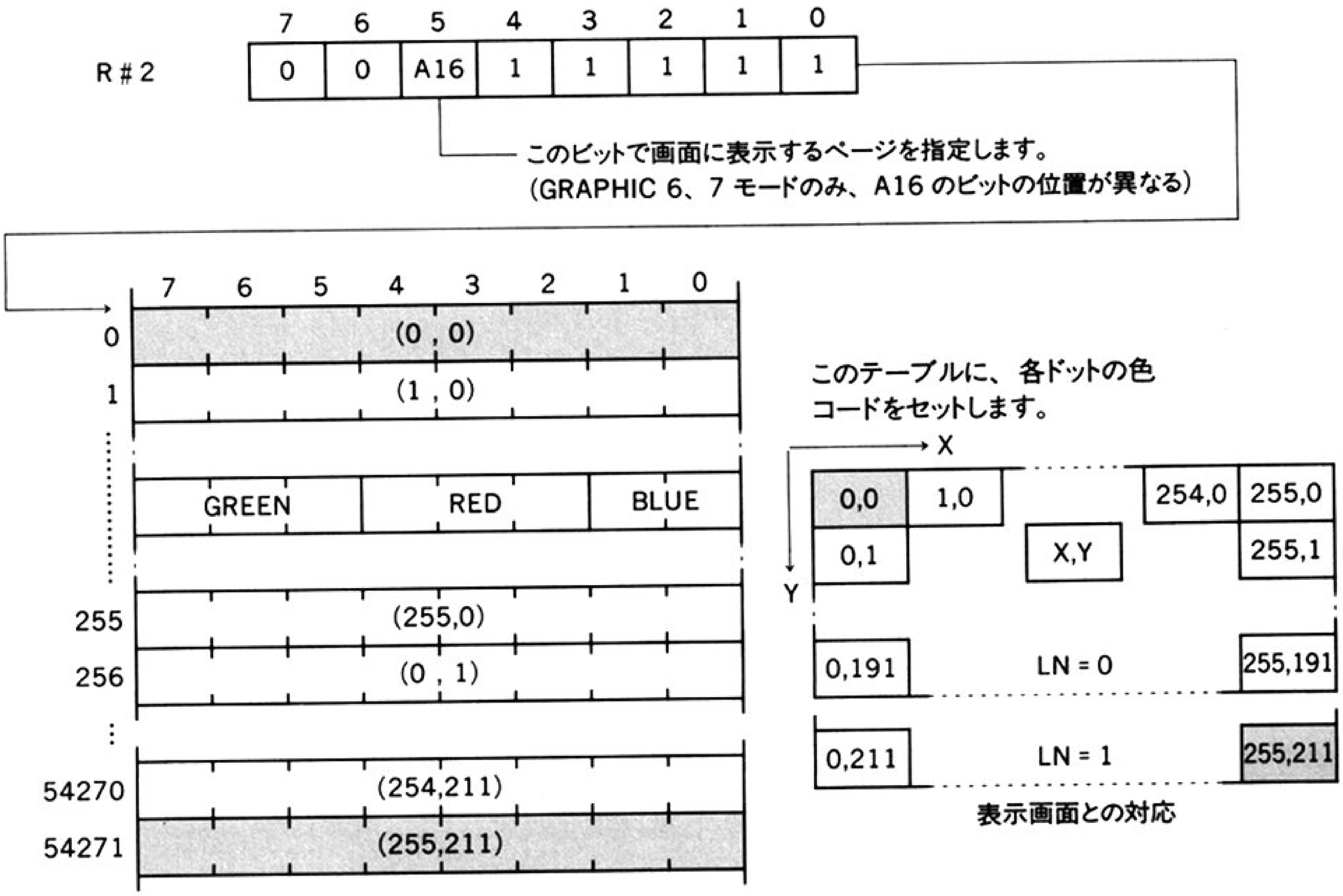
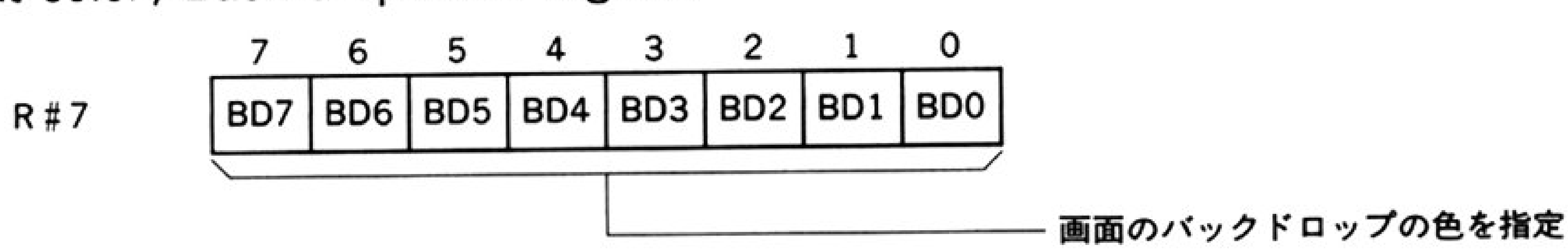


図4.28 GRAPHIC 7 モードのパターンネームテーブル

3. カラーレジスタの設定

R#7で画面のバックドロップの色が指定できます。

Text color / Back drop color register



4. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタR#5とR#11に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタR#6にセットします。詳しくは、「6.2 スプライトモード2」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	1	1	1
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.9.4 GRAPHIC 7 (SCREEN 8)モードの VRAM マップ

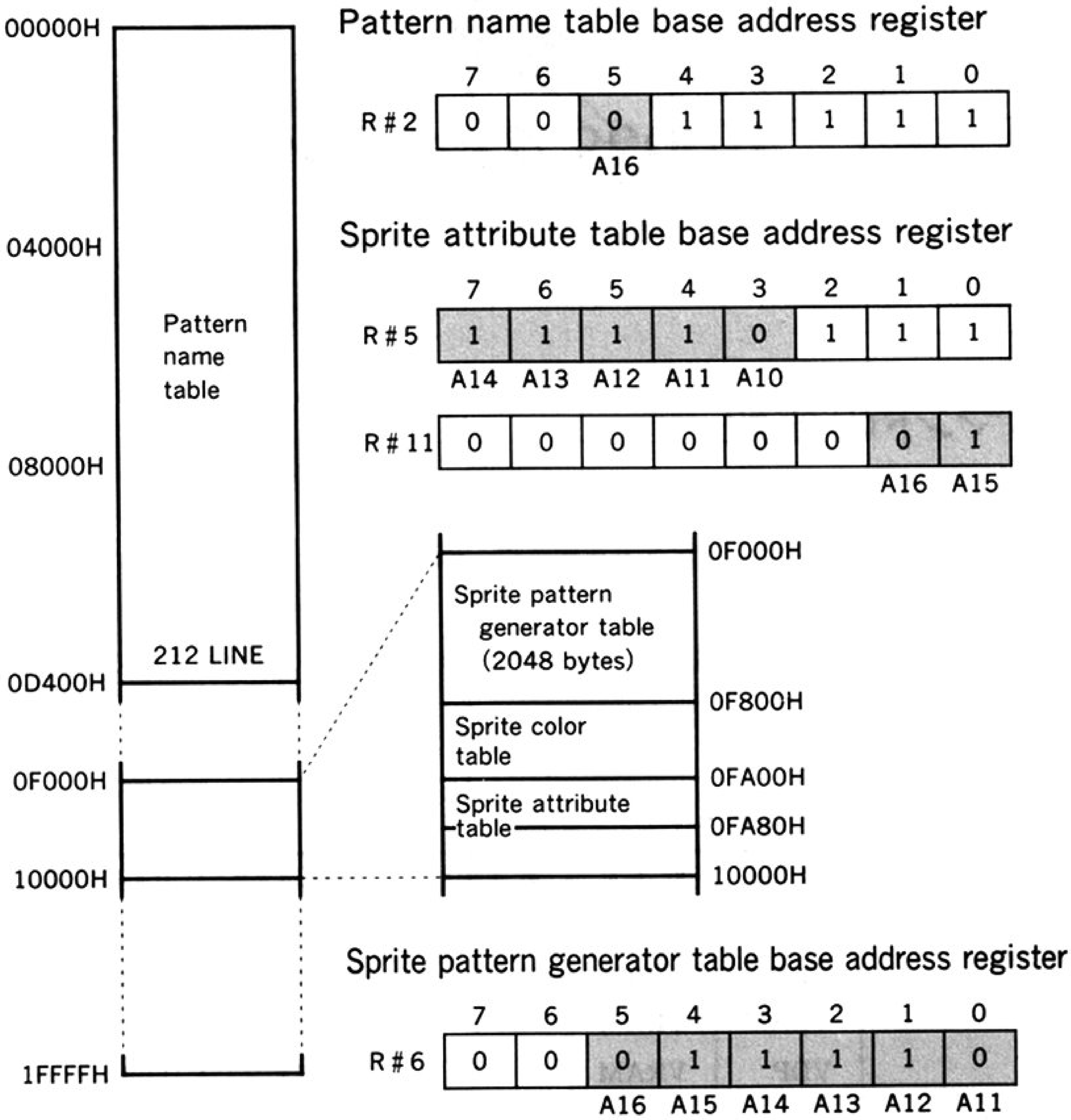


図 4.29 GRAPHIC 7 モードの VRAM マップ

以下同様に、2 ページまで割り付け可能（VRAM　128K 実装機種）です。BASIC では SET PAGE 命令で切り換えて表示することができます。

この画面モードでは 0FA80H から 32 バイトがパレットテーブルとして使われます（MSX のシステムソフトウェアが設定する）。

5章 V9938のコマンド

5.1 コマンドの種類

V9938に備わっている以下のコマンドを使えば、描画(LINE、PSETなど)や画面の部分転送が簡単にできます。

VDPコマンドの使用例は添付のフロッピーディスクの中に入っていますので、そのサンプルプログラムを参照して下さい。

表4.4 VDPのコマンド一覧表

コマンド名	転送元	転送先	転送単位	ニーモニック	R#46(上位4ビット)
High speed move (高速移動)	CPU	VRAM	バイト	HMMC	1 1 1 1
	VRAM	VRAM	バイト	YMMM	1 1 1 0
	VRAM	VRAM	バイト	HMMM	1 1 0 1
	VDP	VRAM	バイト	HMMV	1 1 0 0
Logical move (論理移動)	CPU	VRAM	ドット	LMMC	1 0 1 1
	VRAM	CPU	ドット	LMCM	1 0 1 0
	VRAM	VRAM	ドット	LMMM	1 0 0 1
	VDP	VRAM	ドット	LMMV	1 0 0 0
Line(描線)	VDP	VRAM	ドット	LINE	0 1 1 1
Search(探索)	VDP	VRAM	ドット	SRCH	0 1 1 0
Pset(描点)	VDP	VRAM	ドット	PSET	0 1 0 1
Point	VRAM	VDP	ドット	POINT	0 1 0 0
Invalid	———	———	———	———	0 0 1 1
Invalid	———	———	———	———	0 0 1 0
Invalid	———	———	———	———	0 0 0 1
Stop	———	———	———	———	0 0 0 0

V9938 はレジスタ R＃46 (Command register) にデータが書き込まれると、ステータスレジスタ S＃2 のビット 0 (CE/Command Execute) を 1 にセットしてからコマンドの実行を開始します。必要なパラメータはコマンド実行前にレジスタ R＃32〜R＃45 にセットされていなければなりません。

コマンドの実行が終了すると CE は 0 になります。コマンドの実行を途中で中断するには、STOP コマンドを実行して下さい。

コマンドの動作は、ビットマップモード (SCREEN 5〜8 あるいは GRAPHIC 4〜7) のときしか保証されません。

5.2 ページの概念

V9938 のコマンドを使用する場合の位置パラメータは、すべて X、Y 座標です。つまり、V9938 内部のコマンドプロセッサは、VRAMの全エリアを表示モードに応じたX、Y座標でアクセスします。

画面に表示されるのは、同一ページ内の 212 ラインです (R＃23 で指定)。表示ページの選択は R＃2 で行います。コマンドの実行は、表示ページの選択とは独立して実行されます。

各表示モードにおける VRAM と座標との関係は図 4.30 のとおりです。

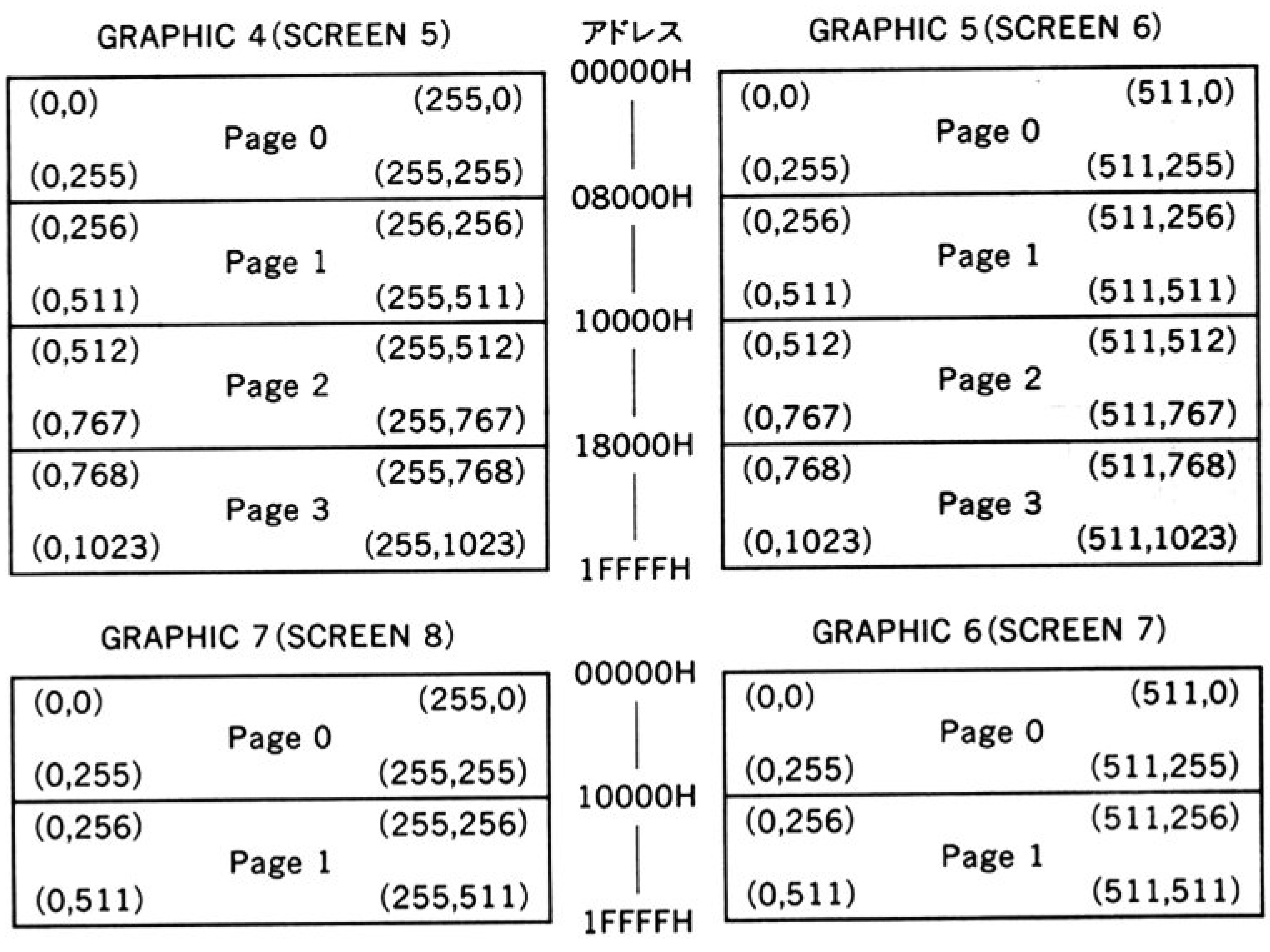


図 4.30　各モードの VRAM の座標

5.3 ロジカルオペレーション

V9938 では、LINE、PSET、LOGICAL　MOVE などのコマンドを実行するときに、指定したデータと画面上の色を演算することができます。演算(ロジカルオペレーション)は、コマンドを指定するときに R # 46 (Command register) の下位 4 ビットに同時に書き込みます。

表 4.5　ロジカルオペレーションコード

演算名	Operation(演算動作)	R # 46(下位 4 ビット)
IMP	DC = SC	0 0 0 0
AND	DC = SC×DC	0 0 0 1
OR	DC = SC+DC	0 0 1 0
EOR	DC = SC×DC+SC×DC	0 0 1 1
NOT	DC = SC	0 1 0 0
――		0 1 0 1
――		0 1 1 0
――		0 1 1 1
TIMP	if SC = 0 then DC = DC else　DC = SC	1 0 0 0
TAND	if SC = 0 then DC = DC else　DC = SC×DC	1 0 0 1
TOR	if SC = 0 then DC = DC else　DC = SC+DC	1 0 1 0
TEOR	if SC = 0 then DC = DC else　DC = SC×DC+SC×DC	1 0 1 1
TNOT	if SC = 0 then DC = DC else　DC = SC	1 1 0 0
――		1 1 0 1
――		1 1 1 0
――		1 1 1 1

* SC = Source Color code
* DC = Destination Color code
* EOR = Exclusive OR

5.4 各コマンドの解説

5.4.1 HMMC(CPU → VRAM 高速転送)

CPU から VRAM の矩形領域(X、Y 座標上)へ、V9938 経由でデータを転送するコマンドです。データの転送は1バイト単位で行われるので、X 座標として指定できる最大値は、表示モードによって制限を受けます。

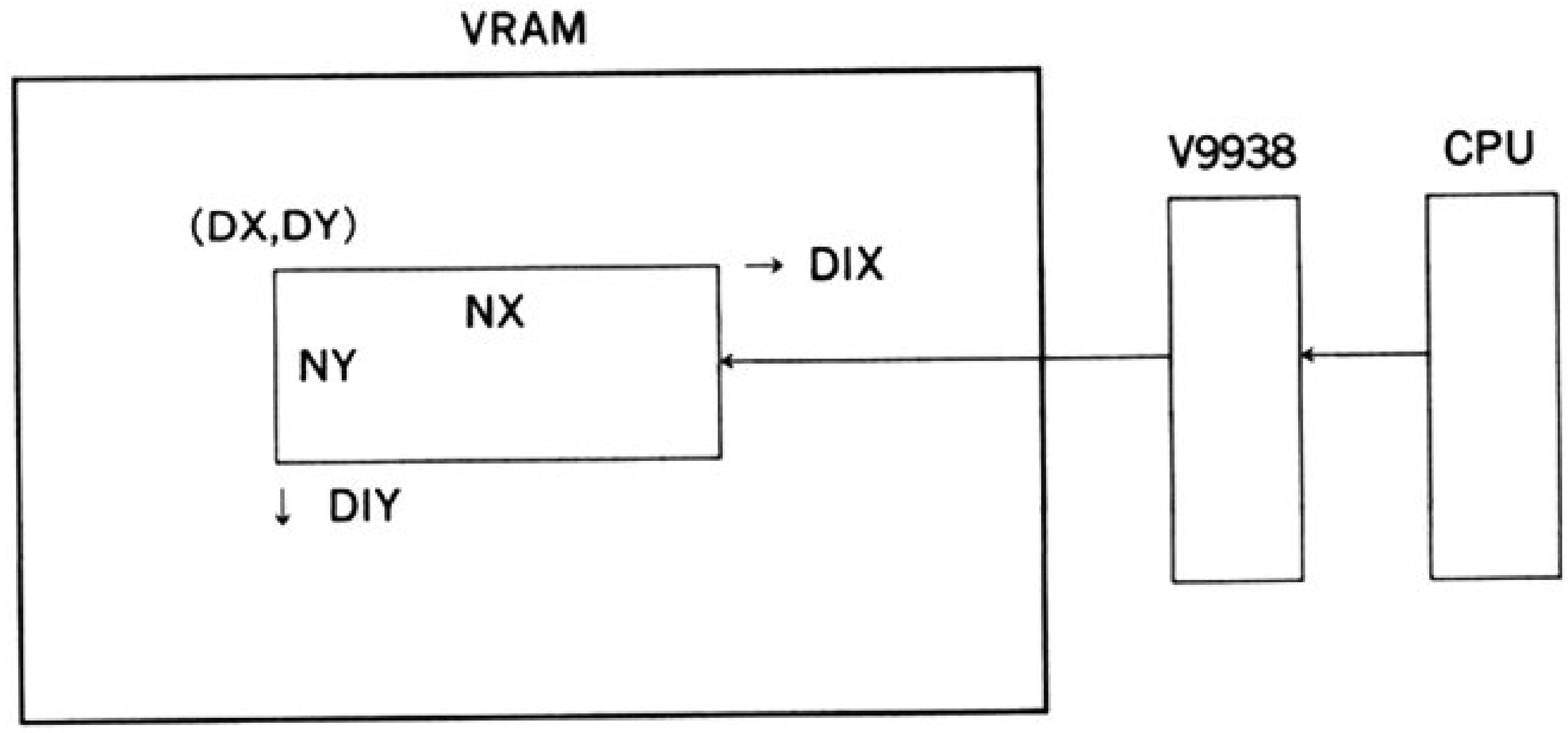


図 4.31　HMMC コマンドの動作

1. HMMC 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

DX	転送先基準点	X 座標(0～511)
DY	転送先基準点	Y 座標(0～1023)
NX	X 方向転送ドット数(1～512*1)	
NY	Y 方向転送ドット数(1～1024*2)	

DX、NX ともに、GRAPHIC 4 と 6 (SCREEN 5 と 7) モードのときは下位1ビット、GRAPHIC 5 (SCREEN 6) モードのときは下位2ビットが無視されます。

*1 512 を指定するときは、NX には「0」を入れます。

*2 1024 を指定するときは、NY には「0」を入れます。

DIX	転送先基準点からの NX の方向(0=右、1=左)
DIY	転送先基準点からの NY の方向(0=下、1=上)

CLR　(R＃44 Color register)　転送データの第1バイト

2) 上記のデータをセットした後、コマンドを実行します。

CMR　(R＃46 Command register) に 11110000B を書き込む

3) ステータスレジスタ S＃2 の TR と CE をチェックしながら、転送データの第2バイト以降を CLR に転送します。

2. HMMC レジスタセットアップ

	7	6	5	4	3	2	1	0	
R＃36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX 転送先基準点
R＃37	0	0	0	0	0	0	0	DX8	
R＃38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY 転送先基準点
R＃39	0	0	0	0	0	0	DY9	DY8	
R＃40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX → X 方向転送ドット数
R＃41	0	0	0	0	0	0	0	NX8	
R＃42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY → Y 方向転送ドット数
R＃43	0	0	0	0	0	0	NY9	NY8	
R＃44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	CLR (SCREEN 5、7) (GRAPHIC 4、6) 転送データ
	X = 2N				X = 2N+1 (N = 0、1、………、127)				
	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	CLR (GRAPHIC 5) (SCREEN 6) 転送データ
	X = 4N		X = 4N+1		X = 4N+2		X = 4N+3 (N = 0、1、………、127)		
	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	CLR (SCREEN 8) (GRAPHIC 7) 転送データ
	1バイト/ドット								
R＃45	0	0	0	0	DIY	DIX	0	0	ARG (Argument register)

DIX：転送方向(X)
DIY：転送方向(Y)

3. HMMC コマンド実行

	7	6	5	4	3	2	1	0	
R＃46	1	1	1	1	0	0	0	0	CMR

4. HMMC 実行フローチャート

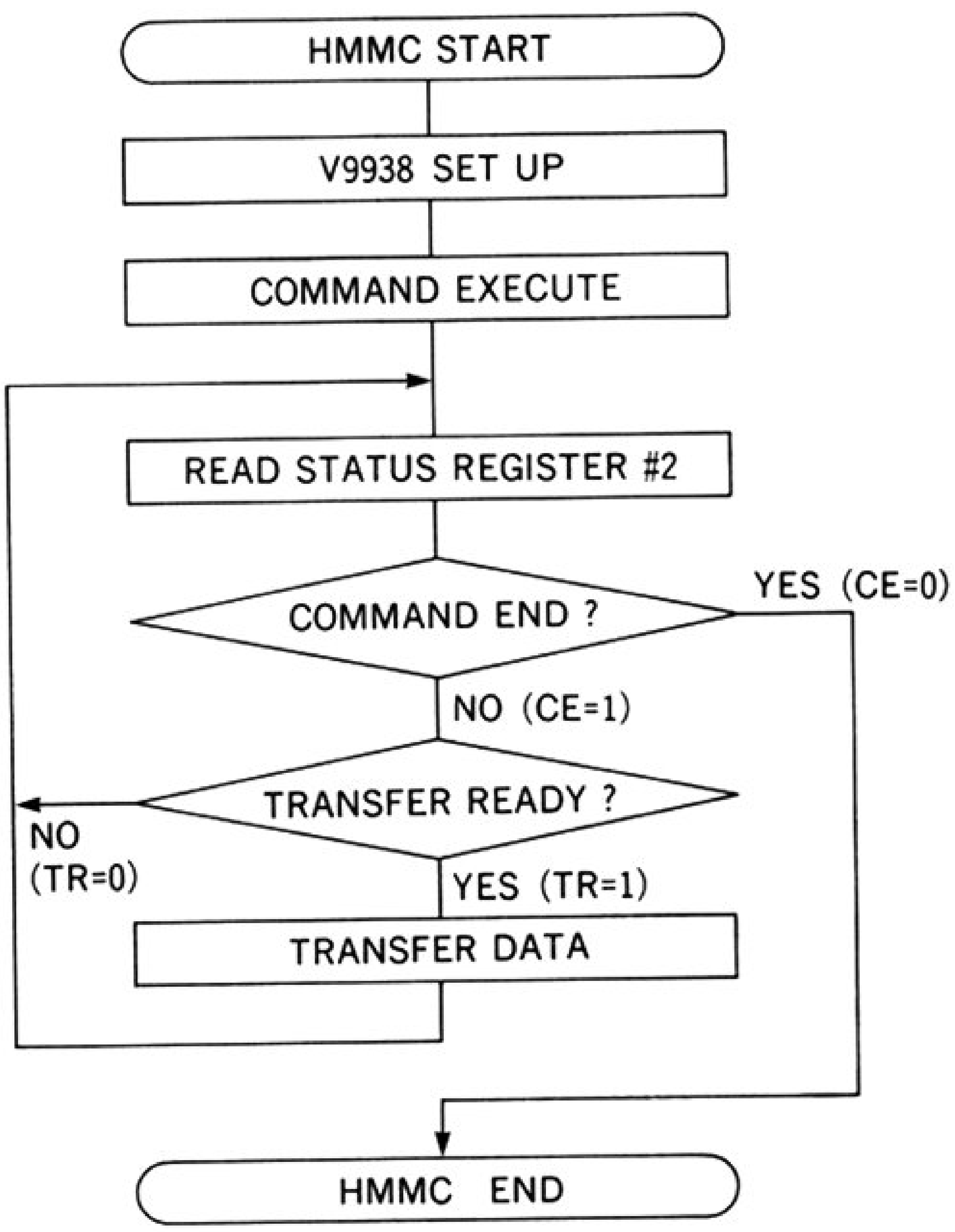


図 4.32　HMMC コマンド実行のフローチャート

5.4.2 YMMM(Y 方向の VRAM 間高速転送)

VRAM 上の DX、SY、NY、DIX、DIY と画面の右端(または左端)で指定される領域を Y 軸方向 (DY で指定) に転送するコマンドです。

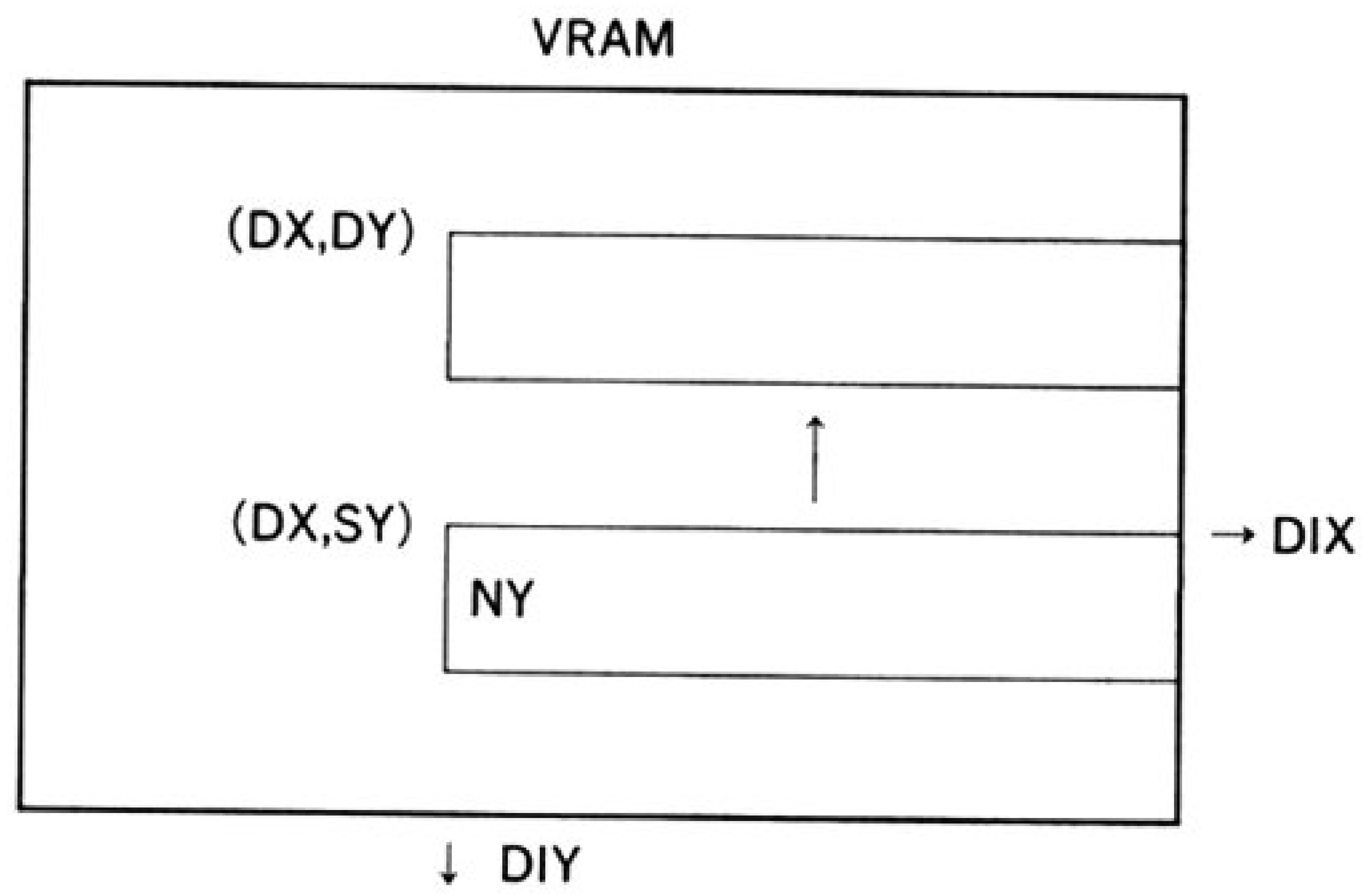


図 4.33　YMMM コマンドの動作

1. YMMM 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

DY	転送先基準点	Y 座標 (0〜1023)
DX	転送元基準点	X 座標 (0〜511)
SY	転送元基準点	Y 座標 (0〜1023)
NY	Y 方向転送ドット数 (1〜1024*)	

DX は GRAPHIC4 と 6 (SCREEN　5 と 7) モードのときは下位 1 ビット、GRAPHIC　5 (SCREEN 6) モードのときは下位 2 ビットが無視されます。

* 1024 を指定するときは、NY には「0」を入れます。

DIX	転送元基準点から左右どちらの画面端までを転送するのかを設定(0 =右、1 =左)
DIY	転送元基準点から見た NY の方向 (0 =下、1 =上)

2) 上記のデータをセットした後コマンドを実行します。

CMR　(R＃46 Command register) に 11100000B を書き込む

3) 以上の操作で V9938 は YMMM コマンドを実行します。コマンドの実行中はステータスレジスタ S＃2 の CE ビットが「1」になり、終了すると「0」になります。

2. YMMM レジスタセットアップ

	7	6	5	4	3	2	1	0	
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY
R#35	0	0	0	0	0	0	SY9	SY8	
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX
R#37	0	0	0	0	0	0	0	DX8	

SY, DX } 転送元基準点

	7	6	5	4	3	2	1	0	
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY →転送先基準点（Y 軸のみ）
R#39	0	0	0	0	0	0	DY9	DY8	
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY → Y 方向転送ドット数
R#43	0	0	0	0	0	0	NY9	NY8	
R#45	0	0	0	0	DIY	DIX	0	0	ARG (Argument register)

DIX ― 転送方向(X)
DIY ― 転送方向(Y)

3. YMMM コマンド実行

	MSB 7	6	5	4	3	2	1	0 LSB	
R#46	1	1	1	0	0	0	0	0	CMR

5.4.3 HMMM(VRAM間高速転送)

VRAMからVRAMへ矩形の領域を転送するコマンドです。データの転送は1バイト単位で行われるので、X座標として指定できる最大値は表示モードによって制限を受けます。

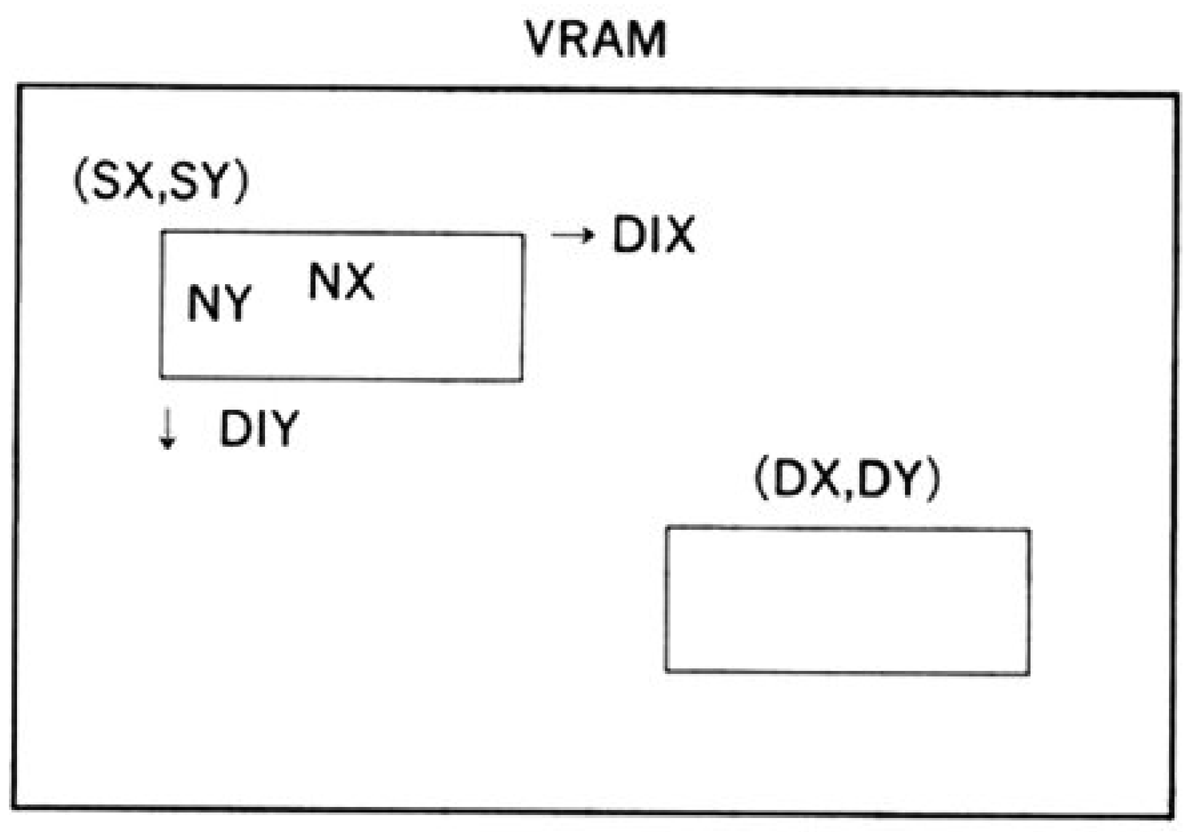


図4.34　HMMMコマンドの動作

1. HMMM実行手順

1) V9938のコマンドレジスタに必要なパラメータをセットします。

SX	転送元基準点　X座標 (0〜511)
SY	転送元基準点　Y座標 (0〜1023)
NX	X方向転送ドット数 (1〜512*1)
NY	Y方向転送ドット数 (1〜1024*2)
DIX	基準点からのNXの方向 (0=右、1=左)
DIY	基準点からのNYの方向 (0=下、1=上)
DX	転送先基準点　X座標 (0〜511)
DY	転送先基準点　Y座標 (0〜1023)

SX、DX、NXともに、GRAPHIC　4と6 (SCREEN　5と7) のときは下位1ビット、GRAPHIC 5 (SCREEN 6) モードのときは下位2ビットが無視されます。

*1 512を指定するときは、NXには「0」を入れます。

*2 1024を指定するときは、NYには「0」を入れます。

2) 上記のデータをセットした後コマンドを実行します。

CMR　(R#46 Command register) に11010000Bを書き込む

3) 以上の操作で V9938 は HMMM コマンドを実行します。コマンドの実行中はステータスレジスタ S#2 の CE ビットが「1」になり、終了すると「0」になります。

2. HMMM レジスタセットアップ

	7	6	5	4	3	2	1	0	
R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	SX 転送元基準点
R#33	0	0	0	0	0	0	0	SX8	
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY 転送元基準点
R#35	0	0	0	0	0	0	SY9	SY8	
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX 転送先基準点
R#37	0	0	0	0	0	0	0	DX8	
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY 転送先基準点
R#39	0	0	0	0	0	0	DY9	DY8	
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX → X 方向転送ドット数
R#41	0	0	0	0	0	0	0	NX8	
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY → Y 方向転送ドット数
R#43	0	0	0	0	0	0	NY9	NY8	
R#45	0	0	0	0	DIY	DIX	0	0	ARG (Argument registers)

DIX — 転送方向(X)
DIY — 転送方向(Y)

3. HMMM コマンド実行

	7	6	5	4	3	2	1	0	
R#46	1	1	0	1	0	0	0	0	CMR

5.4.4 HMMV(長方形の高速塗りつぶし)

VRAMの矩形の領域をカラーコードで塗りつぶすコマンドです。データの転送は1バイト単位で行われるので、X座標として指定できる最大値は表示モードによって制限を受けます。

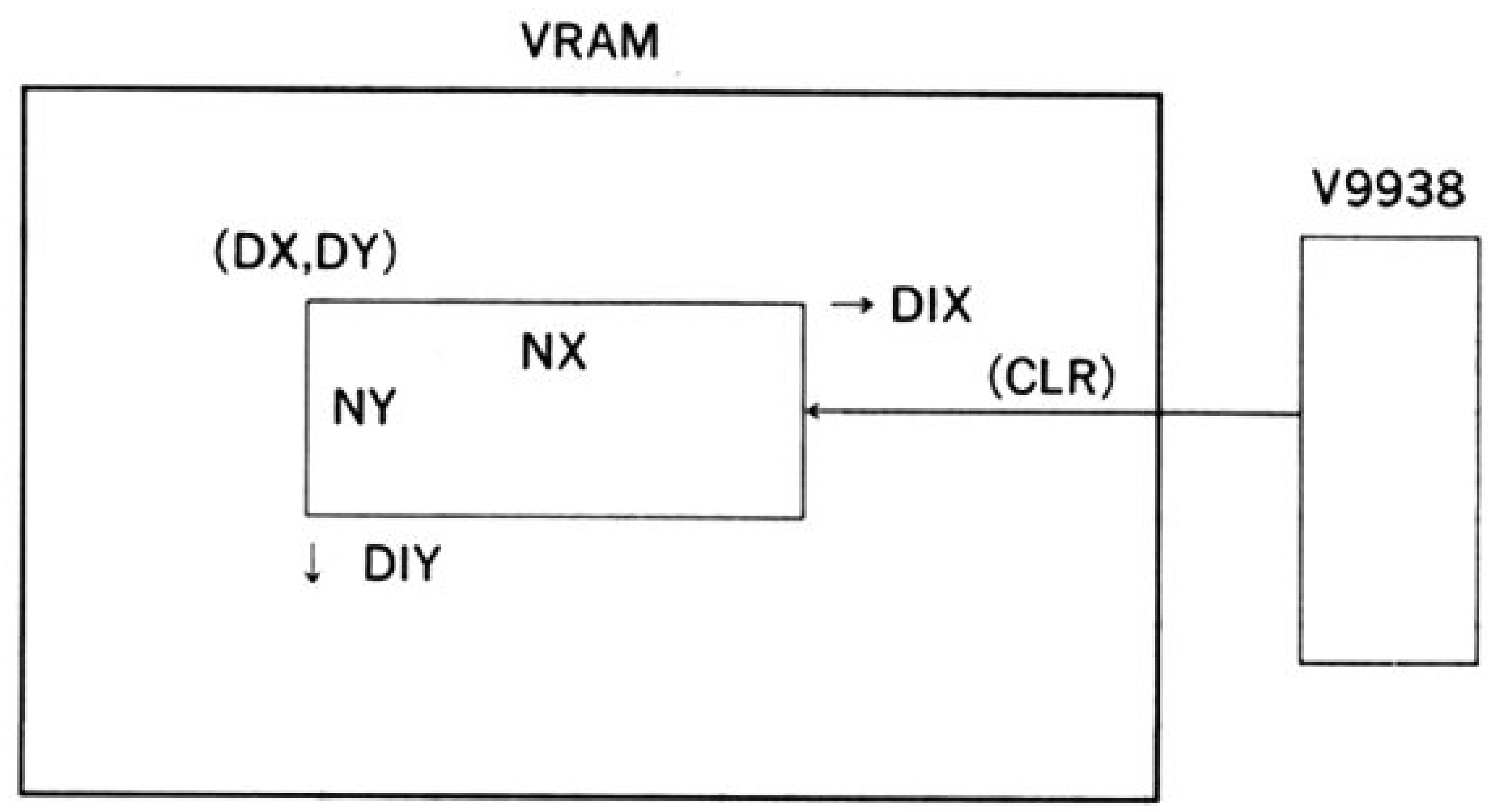


図4.35　HMMVコマンドの動作

1. HMMV実行手順

1) V9938のコマンドレジスタに必要なパラメータをセットします。

NX	X方向転送ドット数 (1～512*1)
NY	Y方向転送ドット数 (1～1024*2)
DIX	転送先基準点からのNXの方向 (0=右、1=左)
DIY	転送先基準点からのNYの方向 (0=下、1=上)
DX	転送先基準点　X座標 (0～511)
DY	転送先基準点　Y座標 (0～1023)

DX、NXともに、GRAPHIC 4と6(SCREEN 5と7)のときは下位1ビット、GRAPHIC 5 (SCREEN 6) モードのときは下位2ビットが無視されます。

*1 512を指定するときは、NXには「0」を入れます。
*2 1024を指定するときは、NYには「0」を入れます。

CLR　　(R#44 Color register)　塗りつぶしデータ

2) 上記のデータをセットした後コマンドを実行します。

CMR　　(R#46 Command register) に11000000Bを書き込む。

3) 以上の操作でV9938はHMMVコマンドを実行します。コマンドの実行中はステータスレジスタS#2のCEビットが「1」になり、終了すると「0」になります。

2. HMMV レジスタセットアップ

	7	6	5	4	3	2	1	0	
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX 転送先基準点
R#37	0	0	0	0	0	0	0	DX8	
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY 転送先基準点
R#39	0	0	0	0	0	0	DY9	DY8	
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX → X方向転送ドット数
R#41	0	0	0	0	0	0	0	NX8	
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY → Y方向転送ドット数
R#43	0	0	0	0	0	0	NY9	NY8	

転送データ (R#44):

	7	6	5	4	3	2	1	0	
R#44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	CLR (SCREEN 5、7) (GRAPHIC 4、6)
	X=2N				X=2N+1 (N=0、1、………、127)				
	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	CLR (SCREEN 6) (GRAPHIC 5)
	X=4N		X=4N+1		X=4N+2		X=4N+3 (N=0、1、………、127)		
	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	CLR (SCREEN 8) (GRAPHIC 7)
	1バイト/ドット								

	7	6	5	4	3	2	1	0	
R#45	0	0	0	0	DIY	DIX	0	0	ARG (Argument register)

DIX — 転送方向(X)
DIY — 転送方向(Y)

3. HMMC コマンド実行

	7	6	5	4	3	2	1	0	
R#46	1	1	0	0	0	0	0	0	CMR

5.4.5 LMMC(CPU → VRAM 論理転送)

CPU から VRAM の矩形領域へ、V9938 経由でデータを転送するコマンドです。データの転送は1ドット単位で行われ、転送先のデータと演算することができます。

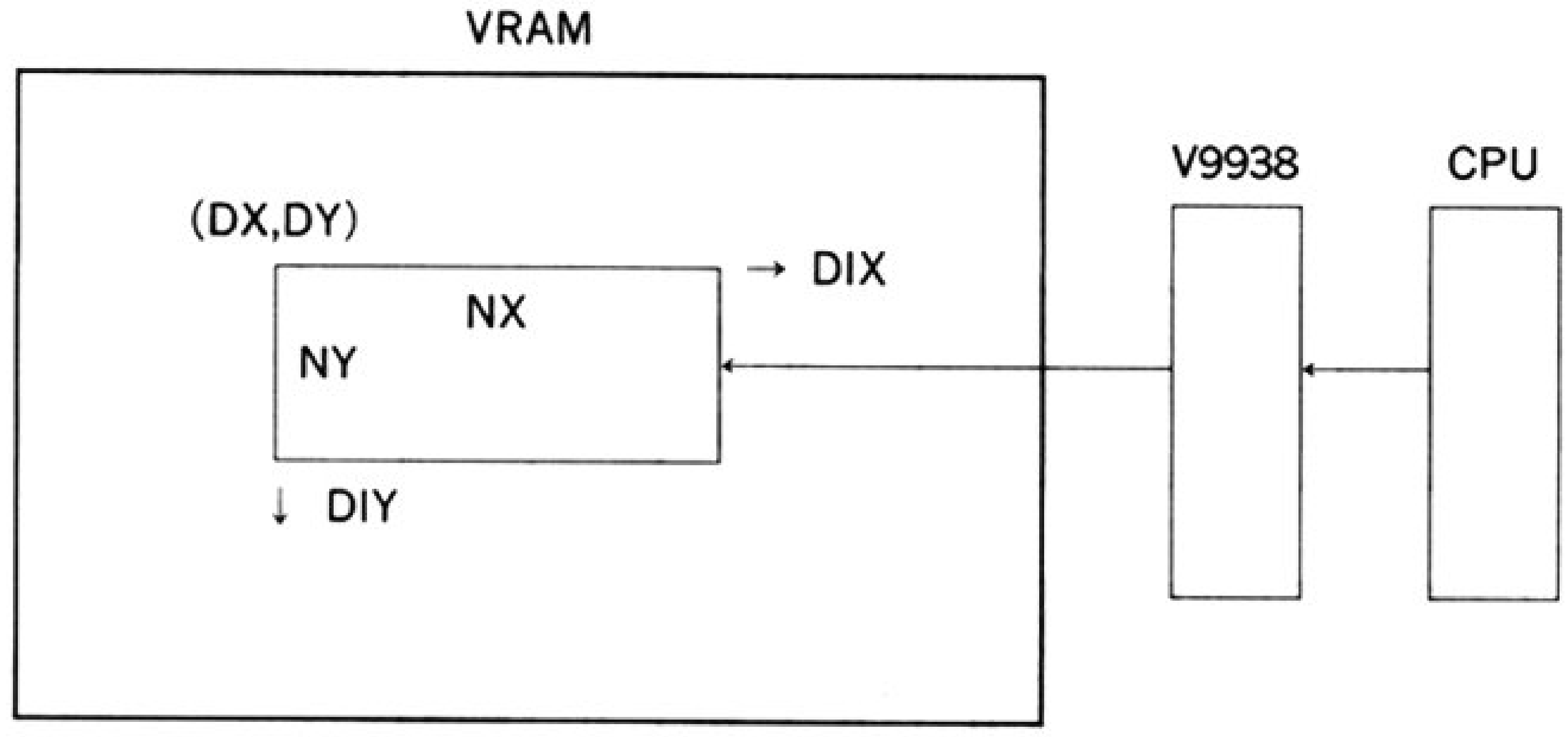


図 4.36　LMMC コマンドの動作

1. LMMC 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

DX	転送先基準点　X 座標 (1〜512*1)
DY	転送先基準点　Y 座標 (1〜1024*2)
NX	X 方向転送ドット数 (0〜511)
NY	Y 方向転送ドット数 (0〜1023)
DIX	転送先基準点からの NX の方向 (0 =右、1 =左)
DIY	転送先基準点からの NY の方向 (0 =下、1 =上)
CLR	(R # 44 Color register) 転送データの第1バイト

*1 512 を指定するときは、NX には「0」を入れます。

*2 1024 を指定するときは、NY には「0」を入れます。

2) 上記のデータをセットした後コマンドを実行します。

CMR	(R # 46 Command register) の上位4ビットに 1011B を、下位4ビットにロジカルオペレーションコード（表 4.5 参照）を書き込む。

3) ステータスレジスタ S # 2 の TR と CE をチェックしながら、転送データの第2バイト以降を CLR レジスタに転送します。

2. LMMC レジスタセットアップ

	7	6	5	4	3	2	1	0	
R # 36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX } 転送先基準点
R # 37	0	0	0	0	0	0	0	DX8	
R # 38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY } 転送先基準点
R # 39	0	0	0	0	0	0	DY9	DY8	
R # 40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX → X 方向転送ドット数
R # 41	0	0	0	0	0	0	0	NX8	
R # 42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY → Y 方向転送ドット数
R # 43	0	0	0	0	0	0	NY9	NY8	
R # 44	0	0	0	0	CR3	CR2	CR1	CR0	CLR (SCREEN 5、7) (GRAPHIC4、6) } 転送データ
	0	0	0	0	0	0	CR1	CR0	CLR (SCREEN 6) (GRAPHIC 5) } 転送データ
	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	CLR (SCREEN 8) (GRAPHIC 7) } 転送データ
R # 45	0	0	0	0	DIY	DIX	0	0	ARG (Argument register)

DIX ── 転送方向(X)
DIY ── 転送方向(Y)

3. LMMC コマンド実行

	7	6	5	4	3	2	1	0	
R # 46	1	0	1	1	LO3	LO2	LO1	LO0	CMR

LO3〜LO0 ── ロジカルオペレーションコード

4. LMMC 実行フローチャート

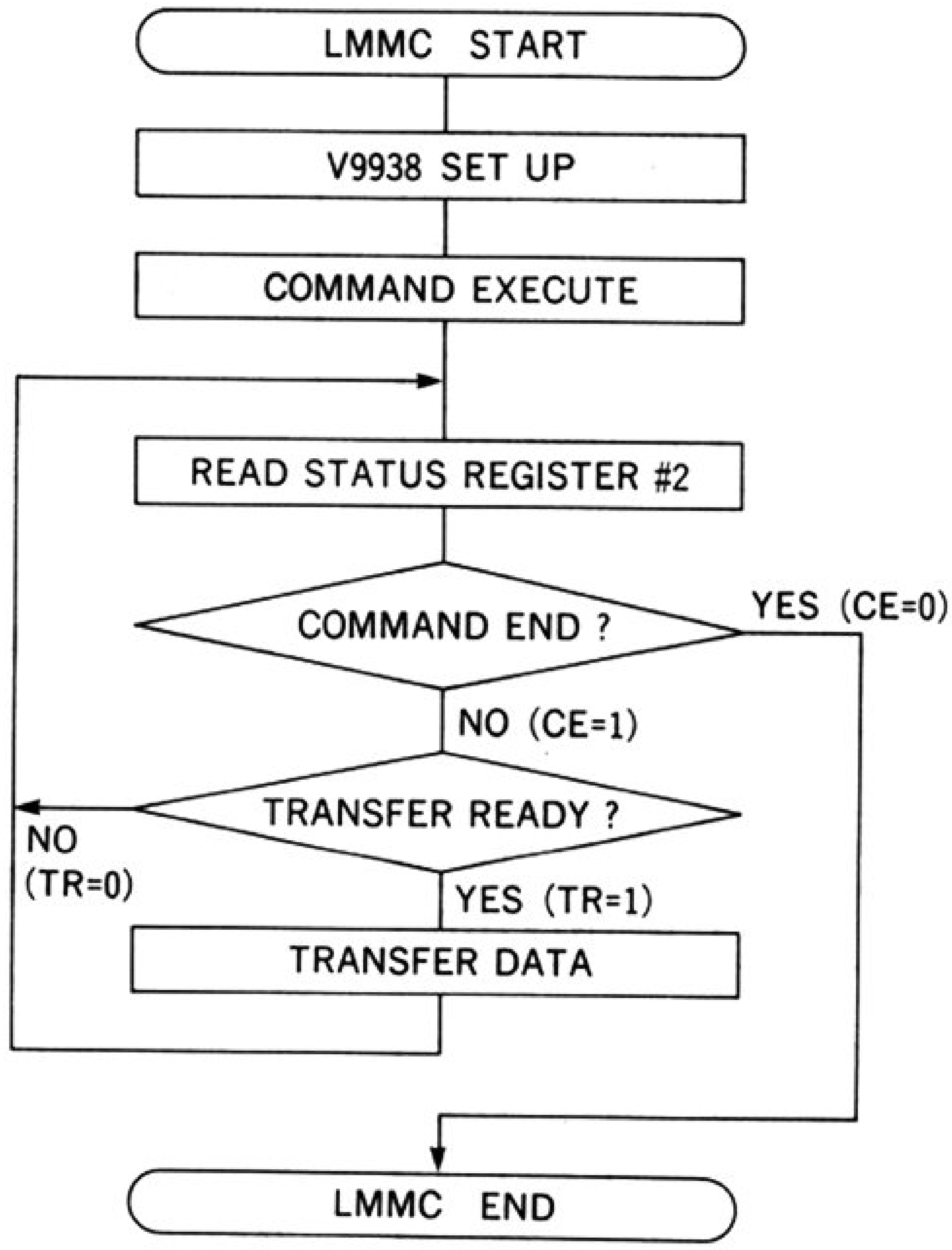


図 4.37 LMMC コマンド実行のフローチャート

5.4.6 LMCM(VRAM → CPU 論理転送)

VRAM の矩形領域（X、Y 座標上）のデータを CPU に転送するコマンドです。データの転送は 1 ドット単位で行われます。

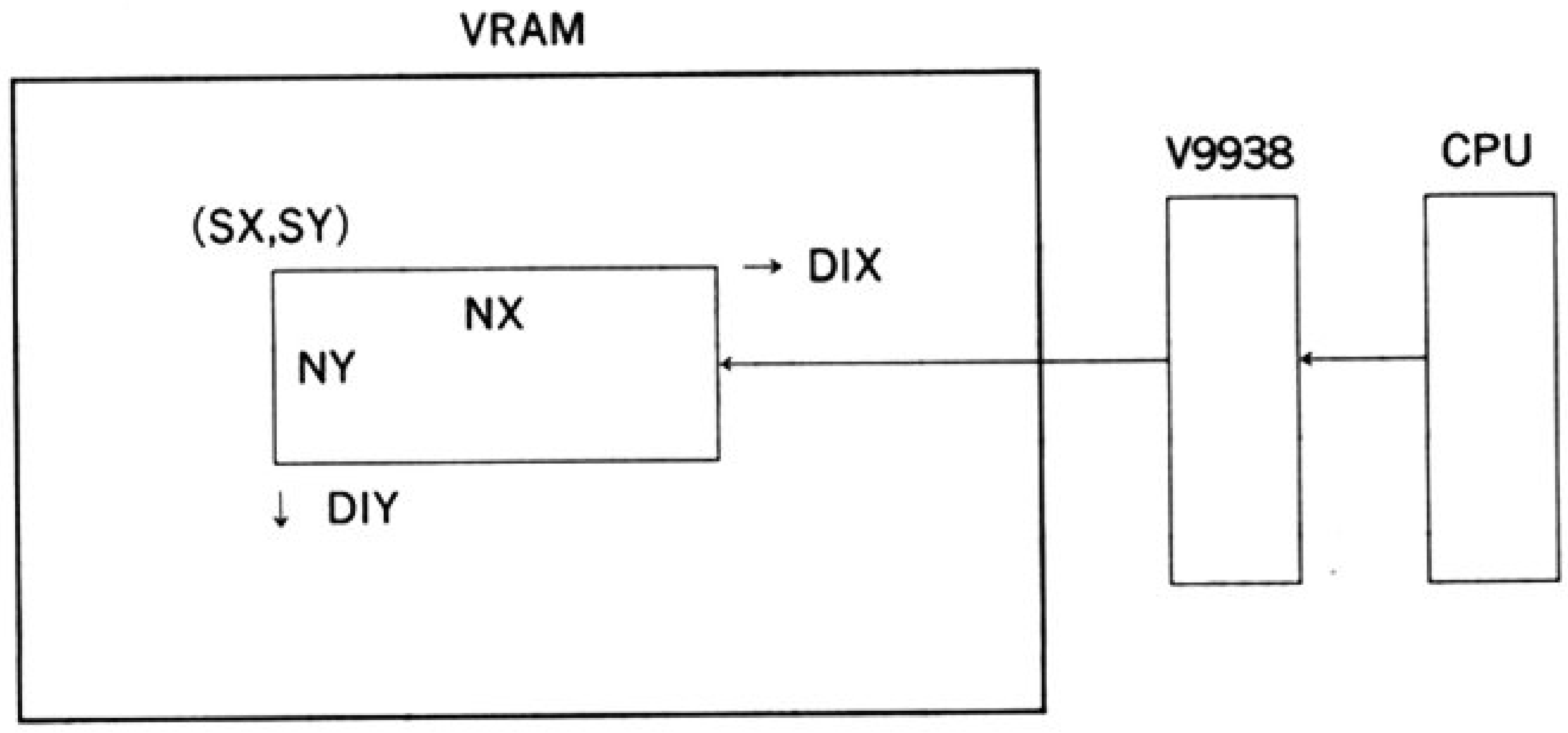


図 4.38　LMCM コマンドの動作

1. LMCM 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

SX	転送元基準点　X 座標（0〜511）	
SY	転送元基準点　Y 座標（0〜1023）	
NX	X 方向転送ドット数（1〜512*1）	
NY	Y 方向転送ドット数（1〜1024*2）	
DIX	転送元基準点からの NX の方向（0 =右、1 =左）	
DIY	転送元基準点からの NX の方向（0 =下、1 =上）	

*1 512 を指定するときは、NX には「0」を入れます。

*2 1024 を指定するときは、NY には「0」を入れます。

2) 上記のデータをセットした後コマンドを実行します。

CMR　　(R # 46 Command register）に 10100000B を書き込む。

3) ステータスレジスタ S#2 の TR と CE をチェックしながら、ステータスレジスタ S#7 を読み出します。

2. LMCM レジスタセットアップ

	7	6	5	4	3	2	1	0	
R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	SX 転送元基準点
R#33	0	0	0	0	0	0	0	SX8	
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SX1	SY0	SY 転送元基準点
R#35	0	0	0	0	0	0	SY9	SY8	
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX → X 方向転送ドット数
R#41	0	0	0	0	0	0	0	NX8	
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY → Y 方向転送ドット数
R#43	0	0	0	0	0	0	NY9	NY8	
R#45	0	0	0	0	DIY	DIX	0	0	ARG (Argument register)

DIX — 転送方向(X)
DIY — 転送方向(Y)

3. LMCM コマンド実行

	7	6	5	4	3	2	1	0	
R#46	1	0	1	0	0	0	0	0	CMR
S#7	0	0	0	0	C3	C2	C1	C0	ステータスレジスタ SCREEN 5、7 (GRAPHIC 4、6)
S#7	0	0	0	0	0	0	C1	C0	ステータスレジスタ SCREEN 6 (GRAPHIC 5)
S#7	C7	C6	C5	C4	C3	C2	C1	C0	ステータスレジスタ SCREEN 8 (GRAPHIC 7)

4. LMCM 実行フローチャート

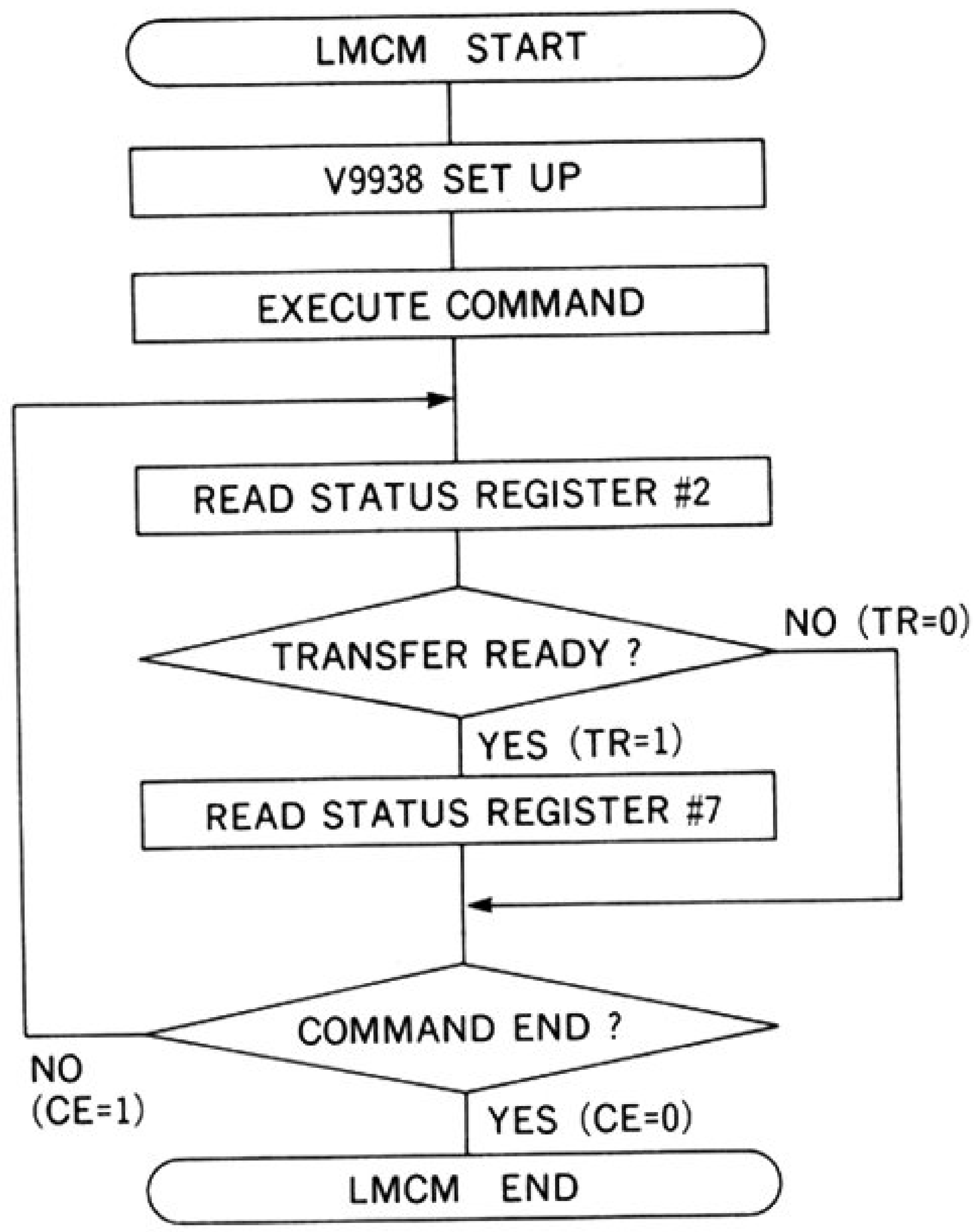


図 4.39 LMCM コマンド実行のフローチャート

注　意

1. 「EXECUTE　COMMAND」の前に TR をリセットする必要があるので、「V9938 SET UP」の中でステータスレジスタ S#7 をリードして下さい。
2. 最後のデータがステータスレジスタ S#7 にセットされて TR = 1 となっていても、V9938 内部ではコマンドは終了し CE = 0 となります。

5.4.7 LMMM(VRAM 間論理転送)

VRAM から VRAM へ矩形の領域を転送するコマンドです。データの転送は1ドット単位で行われ、転送先のデータと論理演算をすることができます。

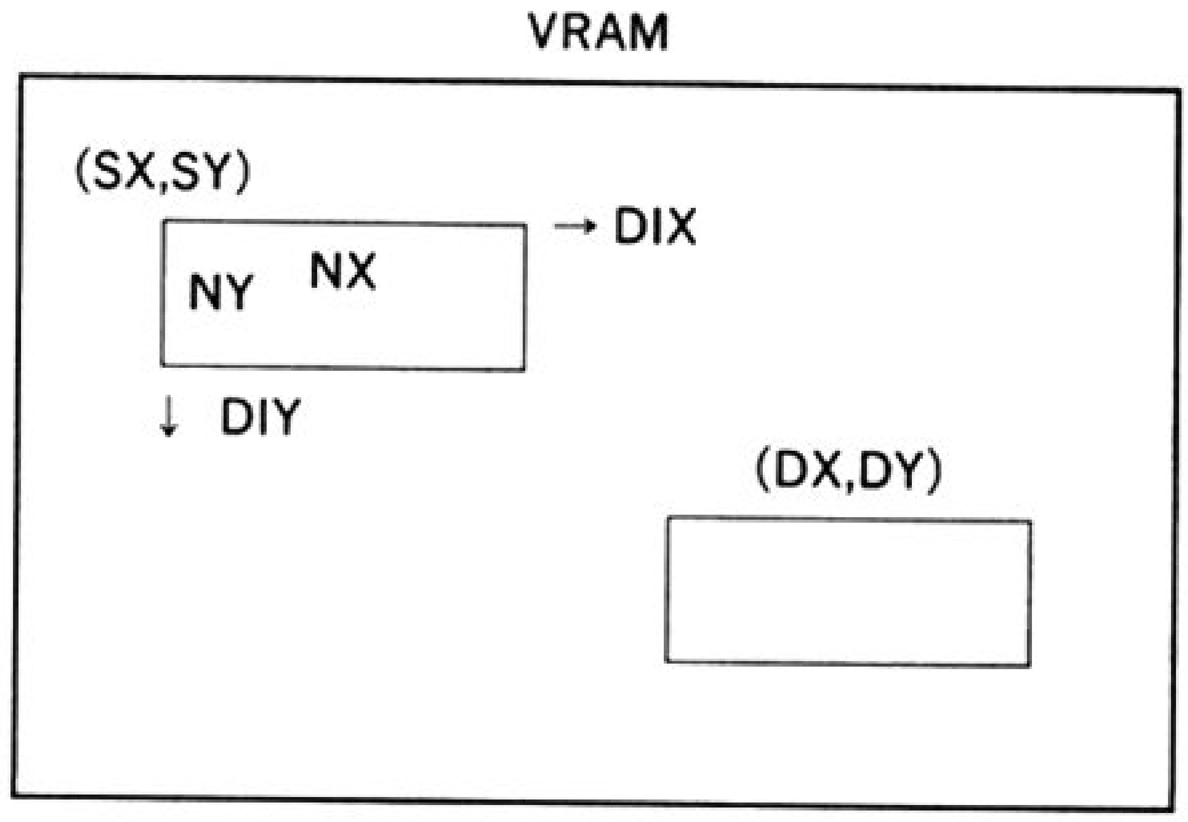


図 4.40　LMMM コマンドの動作

1. LMMM 実行手順

1)　V9938 のコマンドレジスタに必要なパラメータをセットします。

SX	転送元基準点　X 座標（0～511）
SY	転送元基準点　Y 座標（0～1023）
NX	X 方向転送ドット数（1～512*1）
NY	Y 方向転送ドット数（1～1024*2）
DIX	基準点からの NX の方向（0 =右、1 =左）
DIY	基準点からの NY の方向（0 =下、1 =上）
DX	転送先基準点　X 座標（0～511）
DY	転送先基準点　Y 座標（0～1023）

*1 512 を指定するときは、NX には「0」を入れます。
*2 1024 を指定するときは、NY には「0」を入れます。

2)　上記のデータをセットした後コマンドを実行します。

CMR　　(R # 46 Command register) の上位4ビットに 1001B を、下位4ビットにロジカルオペレーションコード（表 4.5 参照）を書き込む。

3) 以上の操作で V9938 は LMMM コマンドを実行します。コマンドの実行中はステータスレジスタ S#2 の CE ビットが「1」になり、終了すると「0」になります。

2. LMMM レジスタセットアップ

	7	6	5	4	3	2	1	0	
R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	SX ⎫ 転送元基準点
R#33	0	0	0	0	0	0	0	SX8	
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY ⎭
R#35	0	0	0	0	0	0	SY9	SY8	
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX ⎫ 転送先基準点
R#37	0	0	0	0	0	0	0	DX8	
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY ⎭
R#39	0	0	0	0	0	0	DY9	DY8	
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX → X 方向転送ドット数
R#41	0	0	0	0	0	0	0	NX8	
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY → Y 方向転送ドット数
R#43	0	0	0	0	0	0	NY9	NY8	
R#45	0	0	0	0	DIY	DIX	0	0	ARG (Argument register)

DIX ── 転送方向(X)
DIY ── 転送方向(Y)

3. LMMM コマンド実行

	7	6	5	4	3	2	1	0	
R#46	1	0	0	1	LO3	LO2	LO1	LO0	CMR

LO3〜LO0 ── ロジカルオペレーションコード

5.4.8 LMMV(VRAM論理塗りつぶし)

VRAMの矩形の領域をカラーコードで塗りつぶすコマンドです。データの転送は1ドット単位で行われ、転送先のデータと論理演算をすることができます。

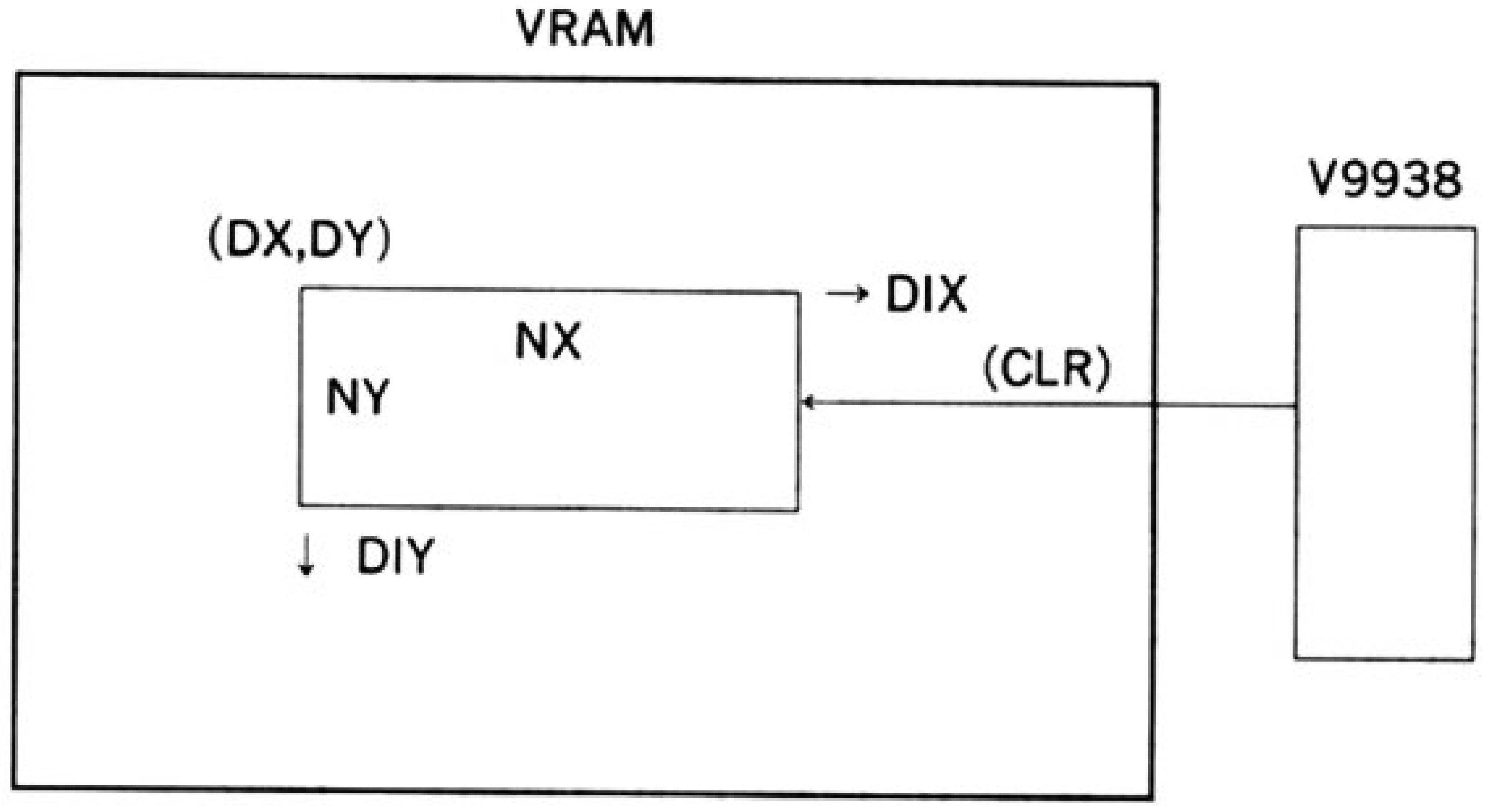


図4.41 LMMVコマンドの動作

1. LMMV実行手順

1) V9938のコマンドレジスタに必要なパラメータをセットします。

NX	X方向転送ドット数 (1〜512*1)
NY	Y方向転送ドット数 (1〜1024*2)
DIX	転送先基準点からのNXの方向 (0=右、1=左)
DIY	転送先基準点からのNYの方向 (0=下、1=上)
DX	転送先基準点　X座標 (0〜511)
DY	転送先基準点　Y座標 (0〜1023)
CLR	(R#44 Color register)　塗りつぶしデータ

*1 512を指定するときは、NXには「0」を入れます。

*2 1024を指定するときは、NYには「0」を入れます。

2) 上記のデータをセットした後コマンドを実行します。

CMR	(R#46 Command register)の上位4ビットに1000Bを、下位4ビットにロジカルオペレーションコード(表4.5参照)を書き込む

3) 以上の操作でV9938はLMMVコマンドを実行します。コマンドの実行中はステータスレジスタS#2のCEビットが「1」になり、終了すると「0」になります。

2. LMMV レジスタセットアップ

	7	6	5	4	3	2	1	0	
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX（転送先基準点）
R#37	0	0	0	0	0	0	0	DX8	
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY（転送先基準点）
R#39	0	0	0	0	0	0	DY9	DY8	
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	NX → X 方向転送ドット数
R#41	0	0	0	0	0	0	0	NX8	
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	NY → Y 方向転送ドット数
R#43	0	0	0	0	0	0	NY9	NY8	
R#44	0	0	0	0	C3	C2	C1	C0	CLR (SCREEN 5、7) (GRAPHIC 4、6)（転送データ）
	0	0	0	0	0	0	C1	C0	CLR (SCREEN 6) (GRAPHIC 5)（転送データ）
	C7	C6	C5	C4	C3	C2	C1	C0	CLR (SCREEN 8) (GRAPHIC 7)（転送データ）
R#45	0	0	0	0	DIY	DIX	0	0	

DIX：転送方向(X)
DIY：転送方向(Y)

3. LMMV コマンド実行

	MSB 7	6	5	4	3	2	1	0 LSB	
R#46	1	0	0	0	LO3	LO2	LO1	LO0	CMR

LO3～LO0：ロジカルオペレーションコード

5.4.9 LINE(直線の描画)

直線を描画するコマンドです。基準点と長辺・短辺からなる長方形の対角線を描画します。データの転送は 1 ドット単位で行われ、転送先のデータと論理演算をすることができます。

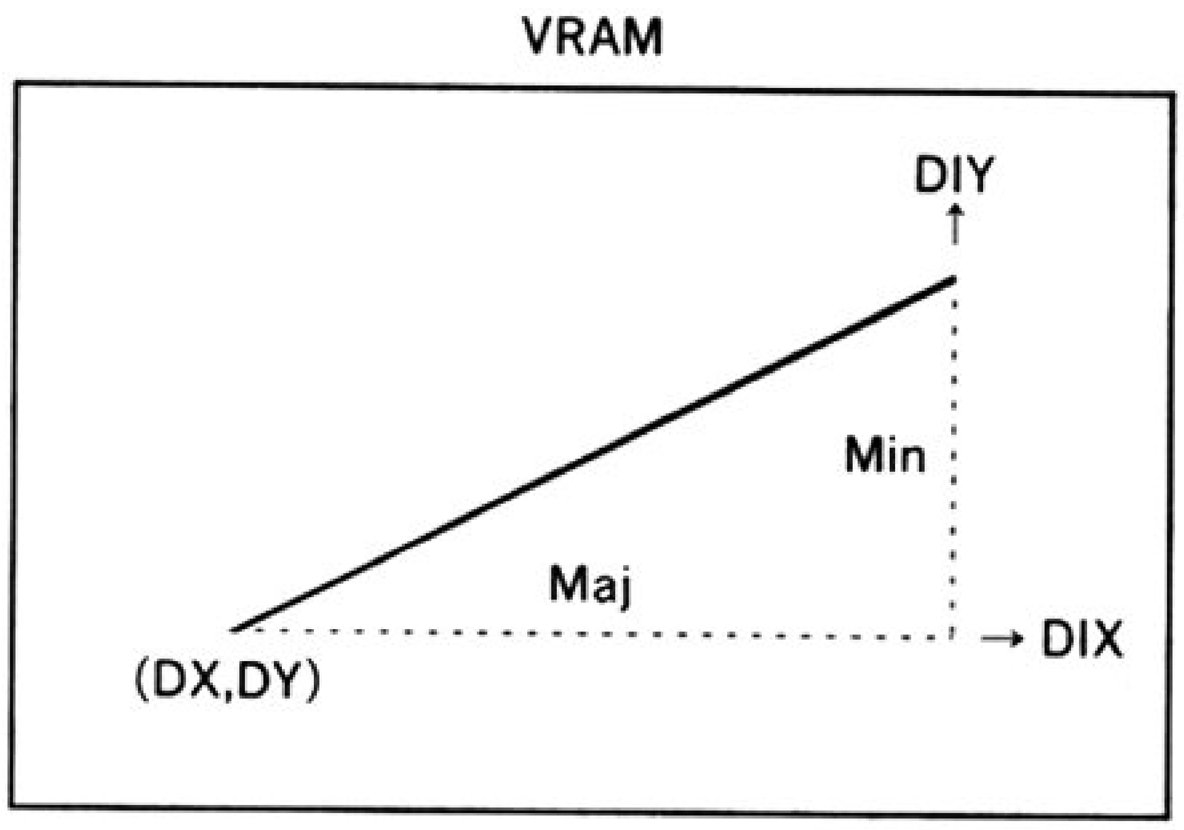


図 4.42　LINE コマンドの動作

1. LINE 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

Maj	長辺ドット数（0～1023)
Min	短辺ドット数（0～511)
MAJ	0 =長辺は X 軸と平行、1 =長辺は Y 軸と平行 / または長辺=短辺
DIX	基準点からの終点の方向（0 =右、1 =左)
DIY	基準点からの終点の方向（0 =下、1 =上)
DX	基準点　X 座標（0～511)
DY	基準点　Y 座標（0～1023)

2) 上記のデータをセットした後コマンドを実行します。

CMR	(R # 46 Command register) の上位 4 ビットに 0111B を、下位 4 ビットにロジカルオペレーションコード（表 4.5 参照）を書き込む。

3) 以上の操作で V9938 は LINE コマンドを実行します。コマンドの実行中はステータスレジスタ S # 2 の CE ビットが「1」になり、終了すると「0」になります。

2. LINE レジスタセットアップ

	7	6	5	4	3	2	1	0	
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX（基準点）
R#37	0	0	0	0	0	0	0	DX8	
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY（基準点）
R#39	0	0	0	0	0	0	DY9	DY8	
R#40	MJ7	MJ6	MJ5	MJ4	MJ3	MJ2	MJ1	MJ0	Maj (NX)→長辺ドット数
R#41	0	0	0	0	0	0	MJ9	MJ8	
R#42	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0	Min (NY)→短辺ドット数
R#43	0	0	0	0	0	0	0	MI8	
R#44	0	0	0	0	C3	C2	C1	C0	CLR (SCREEN 5、7) (GRAPHIC 4、6)（転送データ）
	0	0	0	0	0	0	C1	C0	CLR (SCREEN 6) (GRAPHIC 5)（転送データ）
	C7	C6	C5	C4	C3	C2	C1	C0	CLR (SCREEN 8) (GRAPHIC 7)（転送データ）
R#45	0	0	0	0	DIY	DIX	0	MAJ	ARG (Argument register)

MAJ ― 長辺選択
DIX ― 終点方向(X)
DIY ― 終点方向(Y)

3. LINE コマンド実行

	MSB 7	6	5	4	3	2	1	0 LSB	
R#46	0	1	1	1	LO3	LO2	LO1	LO0	CMR

LO3～LO0 ― ロジカルオペレーションコード

5.4.10 SRCH(色コードのサーチ)

基準点から右（左）に境界色（または非境界色）をサーチするコマンドです。

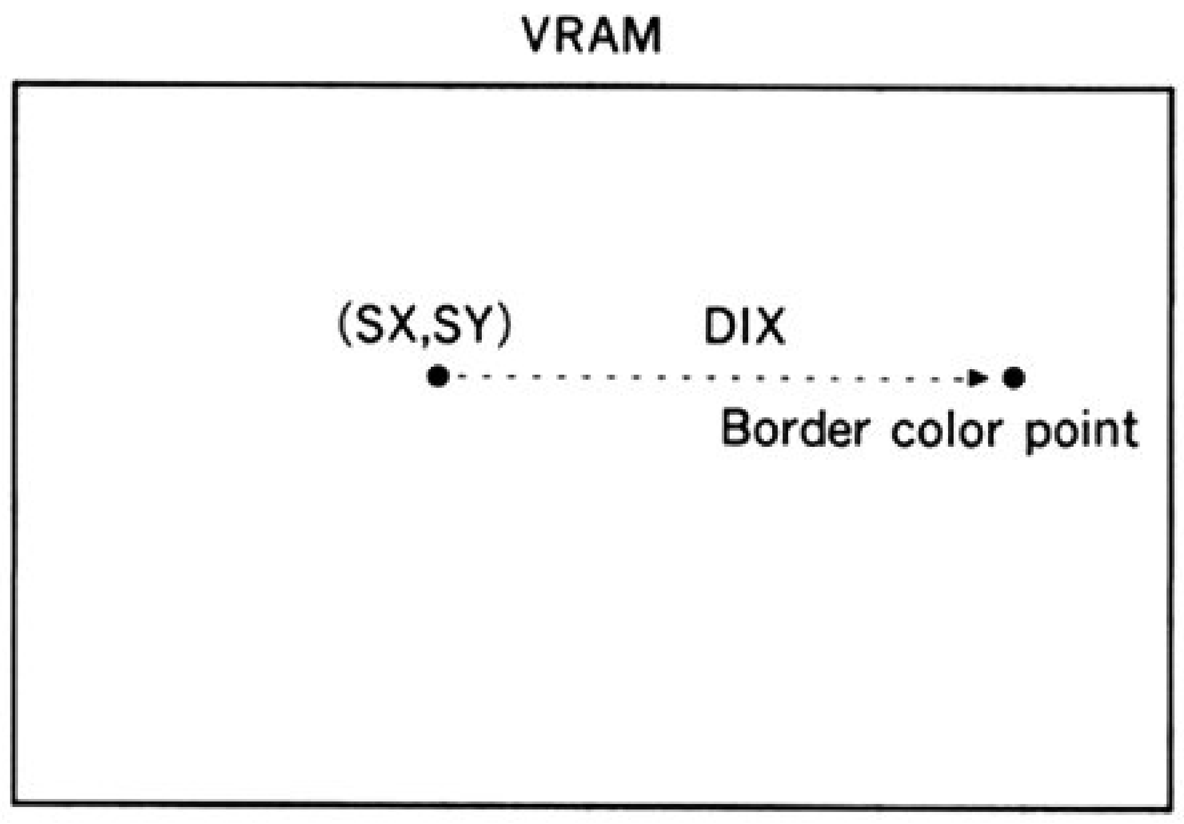


図 4.43　SRCH コマンドの動作

1. SRCH 実行手順

1）V9938 のコマンドレジスタに必要なパラメータをセットします。

DIX	基準点からのサーチ方向（0 =右、1 =左）
EQ	0 =境界色を発見したときに実行を終了、1 =境界色以外を発見したときに実行を終了。
SX	サーチ基準点　X 座標（0～511）
SY	サーチ基準点　Y 座標（0～1023）
CLR	(R # 44 Color register)　境界色

2）上記のデータをセットした後コマンドを実行します。

CMR　(R # 46 Command register）に 01100000B を書き込む。

3）以上の操作で V9938 は SRCH コマンドを実行します。コマンドの実行中はステータスレジスタ S # 2 の CE ビットが「1」になり、終了すると「0」になります。また、画面の端に達するまでに境界色を発見した場合に BD ビットが「1」になります。

2. SRCH レジスタセットアップ

	7	6	5	4	3	2	1	0	
R # 32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	SX サーチ基準点
R # 33	0	0	0	0	0	0	0	SX8	
R # 34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY サーチ基準点
R # 35	0	0	0	0	0	0	SY9	SY8	

	7	6	5	4	3	2	1	0		
R # 44	0	0	0	0	C3	C2	C1	C0	CLR (SCREEN 5、7) (GRAPHIC 4、6)	境界色
	0	0	0	0	0	0	C1	C0	CLR (SCREEN 6) (GRAPHIC 5)	境界色
	C7	C6	C5	C4	C3	C2	C1	C0	CLR (SCREEN 8) (GRAPHIC 7)	境界色

	7	6	5	4	3	2	1	0	
R # 45	0	0	0	0	0	DIX	EQ	0	ARG (Argument register)

EQ —— 実行終了条件
DIX —— サーチ方向(X)

3. SRCH コマンド実行

	7	6	5	4	3	2	1	0	
R # 46	0	1	1	0	0	0	0	0	CMR

	7	6	5	4	3	2	1	0
S # 2	-	-	-	BD	1	1	-	CE

CE —— コマンド終了時 0
BD —— 境界色発見時 1

	7	6	5	4	3	2	1	0	
S # 8	BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0	境界色発見 X 座標
S # 9	1	1	1	1	1	1	BX9	BX8	

4. SRCH 実行フローチャート

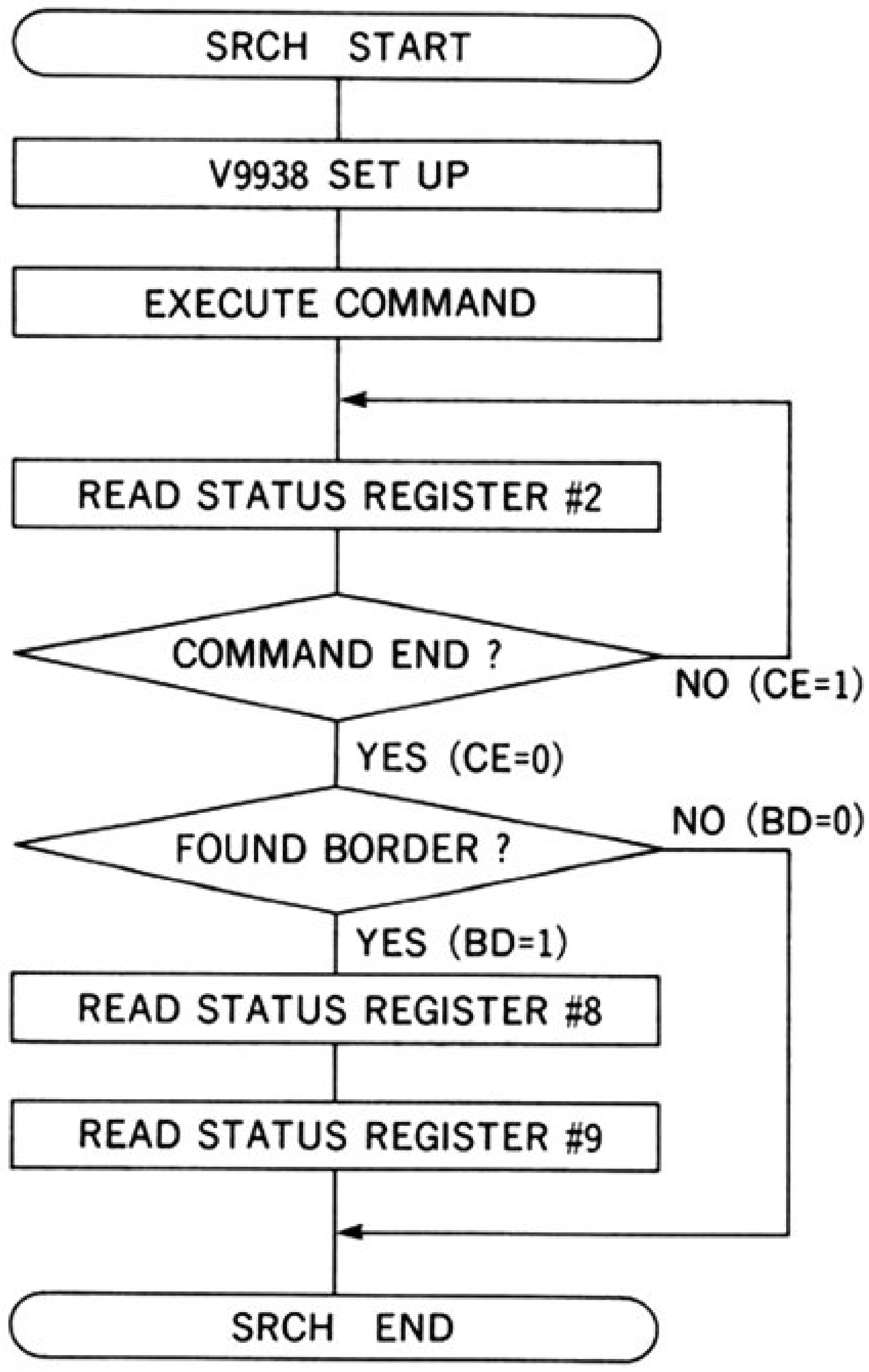


図 4.44 SRCH コマンド実行のフローチャート

5.4.11 PSET(点の描画)

点を描画するコマンドです。すでに表示されている点のデータと論理演算をすることができます。

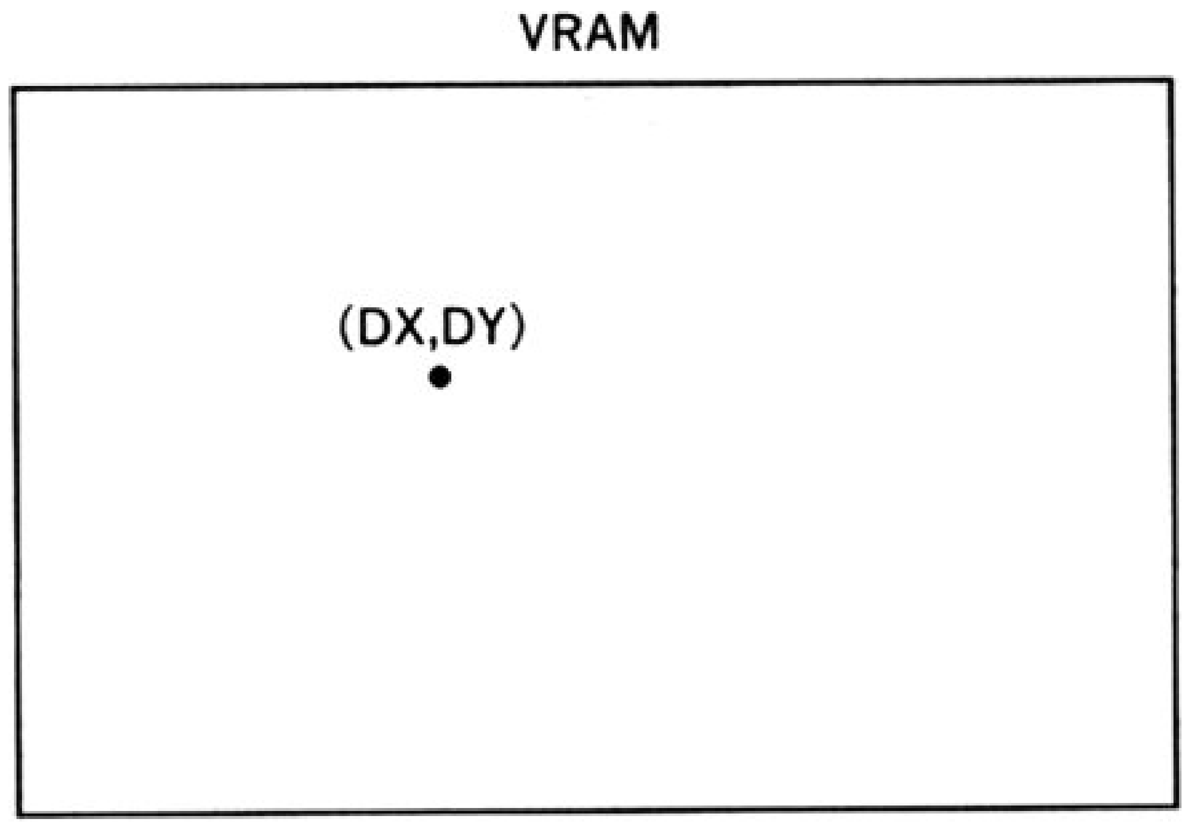


図 4.45　PSET コマンドの動作

1. PSET 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

DX	基準点	X 座標（0〜511）
DY	基準点	Y 座標（0〜1023）

2) 上記のデータをセットした後コマンドを実行します。

CMR　(R # 46 Command register) の上位 4 ビットに 0101B を、下位 4 ビットにロジカルオペレーションコード（表 4.5 参照）を書き込む。

3) 以上の操作で V9938 は PSET コマンドを実行します。コマンドの実行中はステータスレジスタ S # 2 の CE ビットが「1」になり、終了すると「0」になります。

2. PSET レジスタセットアップ

	7	6	5	4	3	2	1	0	
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	DX } 基準点
R#37	0	0	0	0	0	0	0	DX8	
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	DY } 基準点
R#39	0	0	0	0	0	0	DY9	DY8	
R#44	0	0	0	0	C3	C2	C1	C0	CLR (SCREEN 5、7) (GRAPHIC 4、6) } PSET データ
	0	0	0	0	0	0	C1	C0	CLR (SCREEN 6) (GRAPHIC 5) } PSET データ
	C7	C6	C5	C4	C3	C2	C1	C0	CLR (SCREEN 8) (GRAPHIC 7) } PSET データ
R#45	0	0	0	0	0	0	0	0	ARG (Argument register)

3. PSET コマンド実行

	7	6	5	4	3	2	1	0	
R#46	0	1	0	1	LO3	LO2	LO1	LO0	CMR

LO3〜LO0 ── ロジカルオペレーションコード

5.4.12 POINT(色コードの読み出し)

VRAM 上の基準点の色コードを読み出すコマンドです。

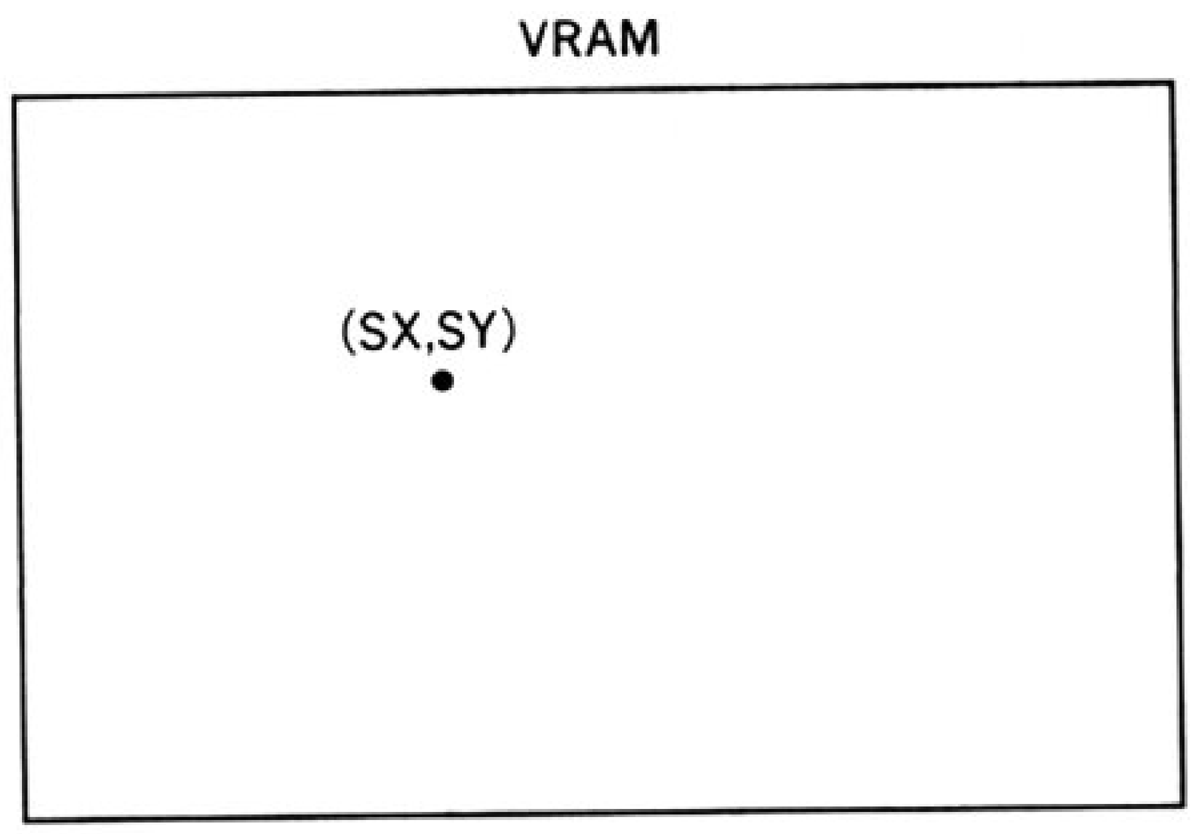


図 4.46　POINT コマンドの動作

1. POINT 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

SX	基準点	X 座標 (0～511)
SY	基準点	Y 座標 (0～1023)

2) 上記のデータをセットした後コマンドを実行します。

CMR　(R＃46 Command register) に 01000000B を書き込む

3) 以上の操作で V9938 は POINT コマンドを実行します。コマンドの実行中はステータスレジスタ S＃2 の CE ビットが「1」になり、終了すると「0」になります。また、読み出された色コードがステータスレジスタ S＃7 にセットされます。

2. POINT レジスタセットアップ

	7	6	5	4	3	2	1	0	
R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	SX 基準点
R#33	0	0	0	0	0	0	0	SX8	
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY 基準点
R#35	0	0	0	0	0	0	SY9	SY8	

3. POINT コマンド実行

	7	6	5	4	3	2	1	0	
R#46	0	1	0	0	0	0	0	0	CMR
S#2	-	-	-	-	1	1	-	CE	CE: コマンド終了時 0
S#7	0	0	0	0	C3	C2	C1	C0	カラーコード SCREEN 5、7 (GRAPHIC 4、6)
	0	0	0	0	0	0	C1	C0	カラーコード SCREEN 6 (GRAPHIC 5)
	C7	C6	C5	C4	C3	C2	C1	C0	カラーコード SCREEN 8 (GRAPHIC 7)

5.5 コマンドの高速化

次の操作をすると、コマンドの処理速度が上がります。

1. スプライトの表示を禁止

レジスタR#8のビット1 (SPD) を「1」にすると、スプライトの処理に使っていた時間をコマンドの実行に使用できるので、コマンドの処理速度が上がります。

2. 画面の表示を禁止

レジスタR#1のビット6 (BL) を「0」にすると、表示の処理に使っていた時間をコマンドの実行に使用できるので、コマンドの処理速度が上がります。

5.6 コマンド終了時のレジスタの状態

コマンド終了時のレジスタの状態は表 4.6 のようになります。

表 4.6　コマンド終了時のレジスタの状態

コマンド名	SX	SY	DX	DY	NX	NY	CLR	CMR H	CLR L	ARG
HMMC	−	−	−	*	−	#	−	0	−	−
YMMM	−	*	−	*	−	#	−	0	−	−
HMMM	−	*	−	*	−	#	−	0	−	−
HMMV	−	−	−	*	−	#	−	0	−	−
LMMC	−	−	−	*	−	#	−	0	−	−
LMCM	−	*	−	−	−	#	*	0	−	−
LMMM	−	*	−	*	−	#	−	0	−	−
LMMV	−	−	−	*	−	#	−	0	−	−
LINE	−	−	−	*	−	−	−	0	−	−
SRCH	−	−	−	−	−	−	−	0	−	−
PSET	−	−	−	−	−	−	−	0	−	−
PINT	−	−	−	−	−	−	*	0	−	−

− 変化なし
＊ コマンド終了時の座標（SY＊、DY＊）とカラーコード
＃ 画面端を検出したときはそのときのカウント数（NYB）となる

SY＊、DY＊、および NYB の値は Y 方向実行ドット数を N とすると、次のように計算されます。

SY＊= SY＋N、DY＊= DY＋N　(DIY = 0)

SY＊= SY−N、DY＊= DY−N　(DIY = 1)

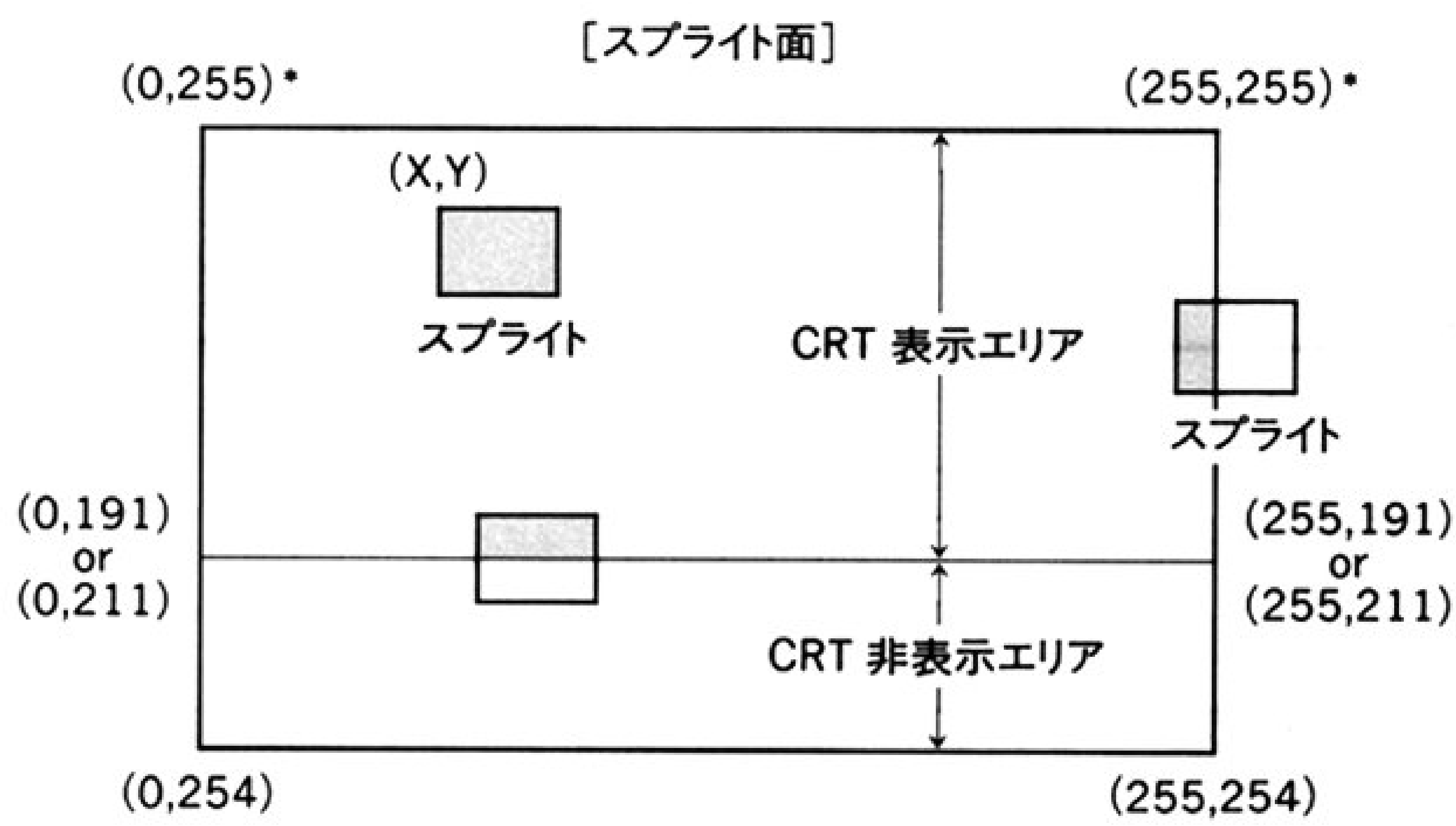
NYB = NY−N

注　意　LINE で MAJ = 0 のときは、N = N−1 となります。

6章 スプライト

V9938は32個のスプライトを表示することができます。スプライトは8×8または16×16ドットの大きさです。また、横方向の1ドットの大きさは表示モードにかかわらず画面の1/256で、画面上の任意の位置に表示することができます。

スプライトは独立した表示面を持っており、他の表示面のデータには影響を及ぼしません。



*スプライトのY座標の最上端の座標は255です。

図4.47 スプライト表示面

V9938には2種類のスプライト表示モードがあります。この2種類のモードは、画面表示モードによって自動的に選択されます。

■スプライトモード1	SCREEN 1～3
	GRAPHIC 1、GRAPHIC 2、MULTI COLOR
■スプライトモード2	SCREEN 4～8
	GRAPHIC 3～7

6.1 スプライトモード1

スプライトモード1はTMS9918のスプライトと同機能で、GRAPHIC 1、2、MULTI COLORモード（SCREEN 1〜3）で使うことができます。

6.1.1 スプライトモード1の特徴

32個のスプライトには各々に#0〜#31までの番号がつけられており、番号の若い方が優先度が高くなります。CRTの1水平線上には優先度の高い順に4個表示され、5番目以降のスプライトは表示されません。

スプライトモード1はTMS9918のスプライトモードと互換性があります。

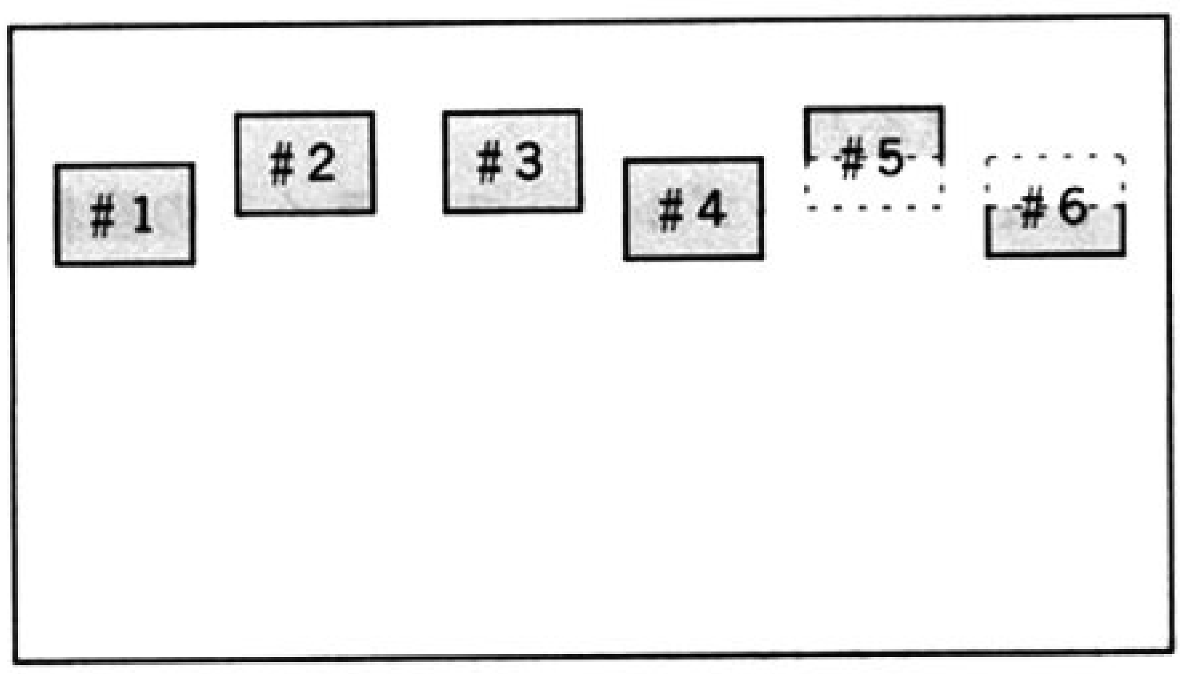


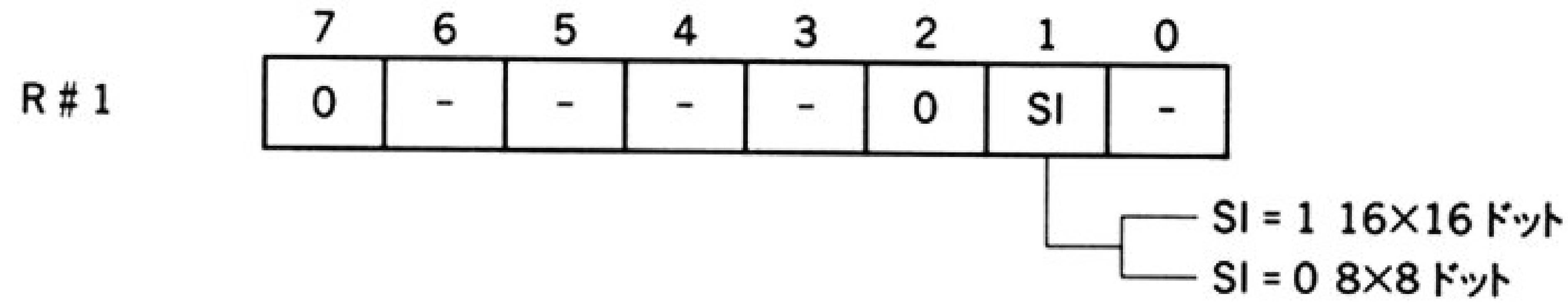
図4.48　スプライトモード1の最大表示数

2個のスプライトが衝突した（パターンの1の部分が重なった）ときは、ステータスレジスタS#0のビット5が「1」になるので、衝突の発生を知ることができます。

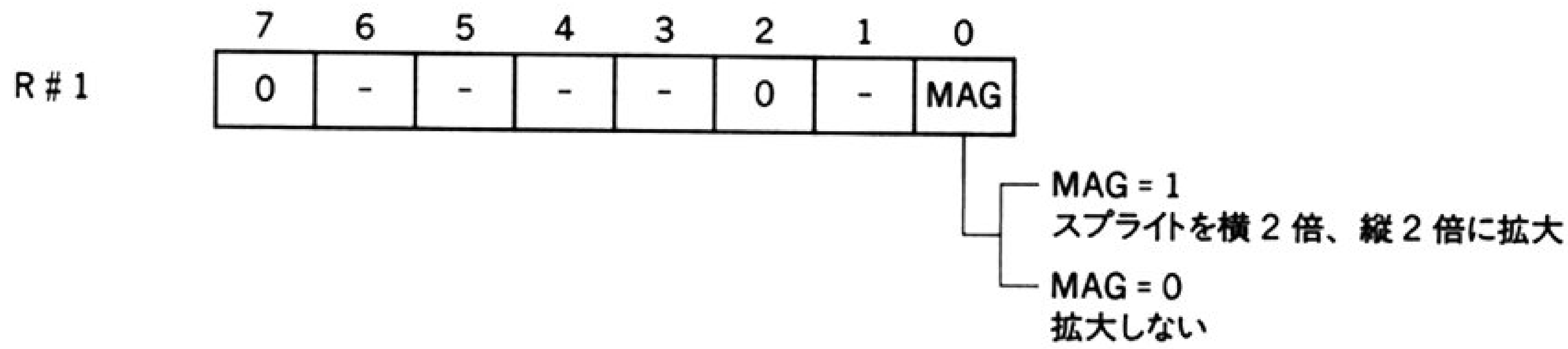
また、1水平線上に5個以上のスプライトが並んだときには、ステータスレジスタS#0のビット6が「1」になり、S#0の下位5ビットに5番目のスプライトの番号がセットされます。

6.1.2 スプライトモード1の表示

スプライトを表示するには、次の操作を行います。



1) スプライトの拡大……R#1のビット0



2) スプライトパターンジェネレータテーブルのセット

VRAM上のスプライトパターンジェネレータテーブルに、スプライトのパターンをセットします（#0～#255）。

3) スプライトアトリビュートテーブルのセット

VRAM上のスプライトアトリビュートテーブルに、スプライトの座標、スプライトのパターン番号と色をセットします（#0～#31）。

6.1.3 スプライトアトリビュートテーブル

32個あるスプライトの各々の表示位置(X、Y)、色、パターン番号などを指定するためのVRAMの領域です。この領域の先頭アドレスはR#5とR#11（Sprite attribute table base address register）で指定します。スプライト1個につき4バイトのデータを書き込みます。

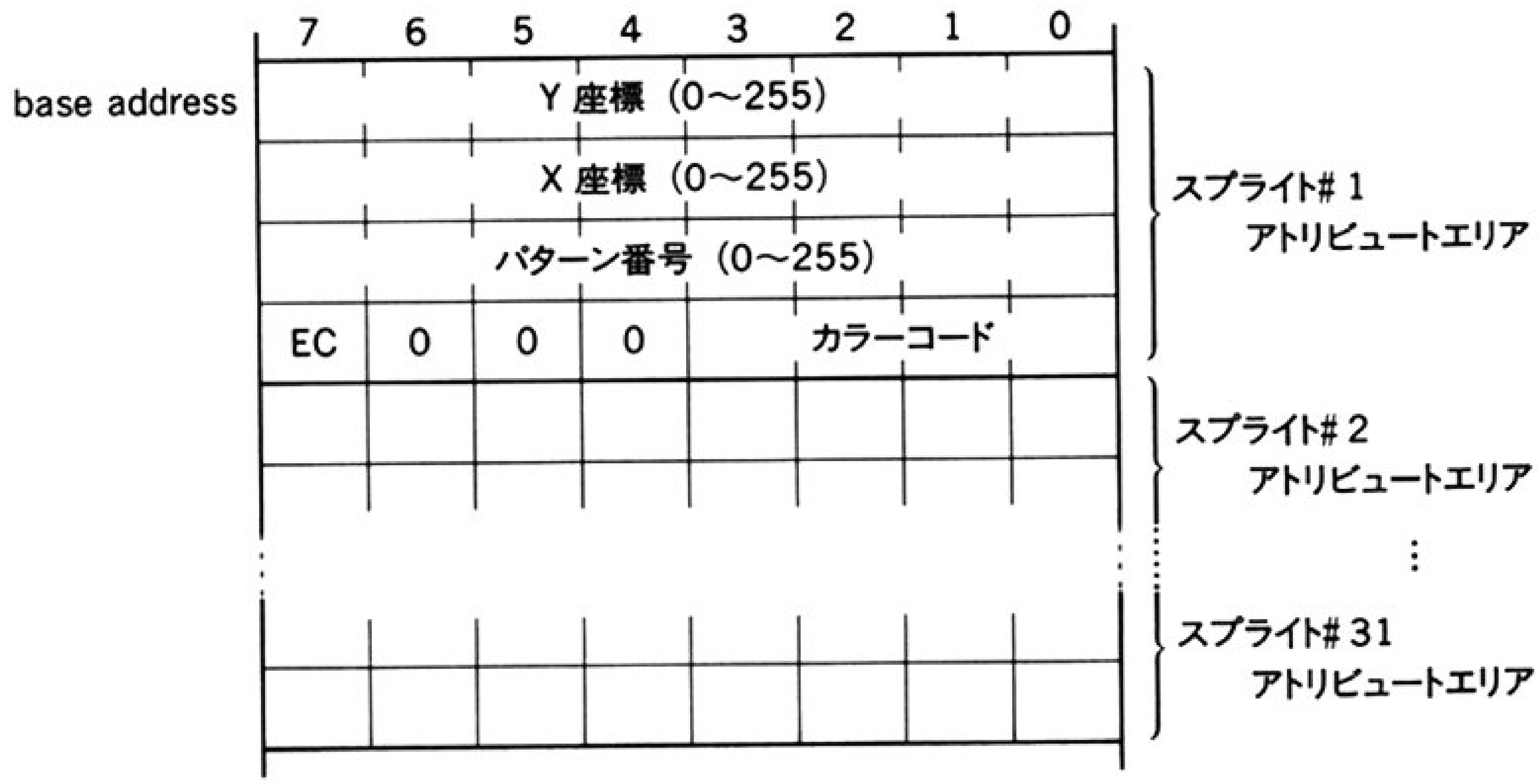


図 4.49 スプライトモード 1 のアトリビュートテーブル

1) Y 座標 (0〜255)

スプライトの Y 座標を指定します。

この Y 座標の値を 208 にすると、そのスプライトより優先度の低いスプライトは表示されません。例えば、# 10 のスプライトの Y 座標を 208 にすると、# 10〜# 31 のスプライトは表示されません。

2) X 座標 (0〜255)

スプライトの X 座標を指定します。

3) パターン番号 (0〜255)

スプライトパターンジェネレータテーブル上のパターン番号を指定します。スプライトのサイズが 16×16 のときは、4 個のパターン番号を 1 個のスプライトに使用しますが、このうちのどれを指定してもかまいません。

スプライトのサイズを 8×8 に指定した場合は 256 個のパターンを、16×16 に指定した場合は 64 個のパターンを指定することができます。

4) カラーコード (0〜15)

スプライトパターンのビットが「1」の部分の色を指定します。「0」の部分は透明です。

5) EC (Early Clock)

このビットが「1」のとき、スプライトの表示は 32 ドット左にシフトされます。この機能を使うと、スプライトを画面の左端から 1 ドットずつ出現させることができます。

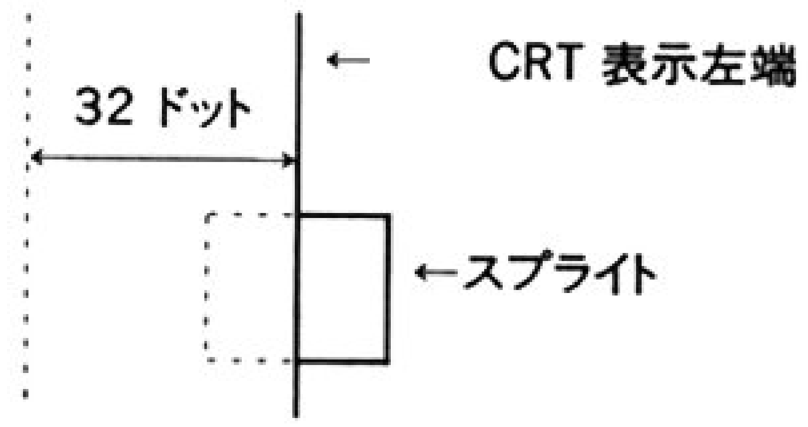


図 4.50 アトリビュートテーブルの EC ビット

6.1.4 スプライトパターンジェネレータテーブル

スプライトのパターン(形)を指定するための VRAM の領域です。この領域の先頭アドレスは R # 6 (Sprite pattern generator table base address register) で指定します。

この領域には、8 バイトで 1 単位のパターンを最大 256 個書き込むことができます。これらのパターンには# 0〜# 255 までの番号がつけられており、スプライトのサイズが 8×8 ドットのときはスプライト 1 個に対して 1 個のパターンが対応し、16×16 ドットのときはスプライト 1 個に対して 4 個のパターンが対応します。

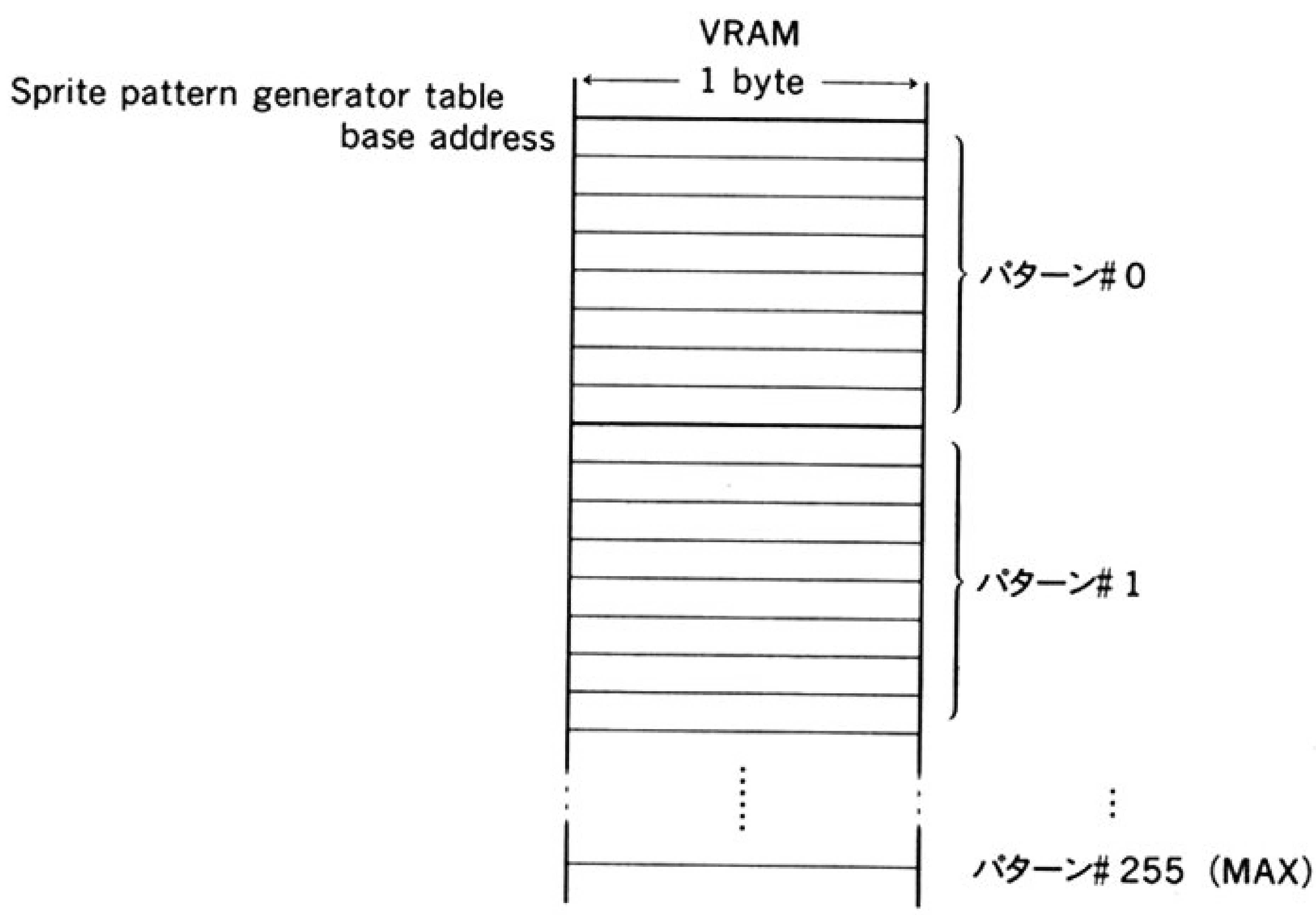


図 4.51 スプライトモード 1 のパターンジェネレータテーブル

6.1.5 スプライトパターンジェネレータテーブルデータセット例

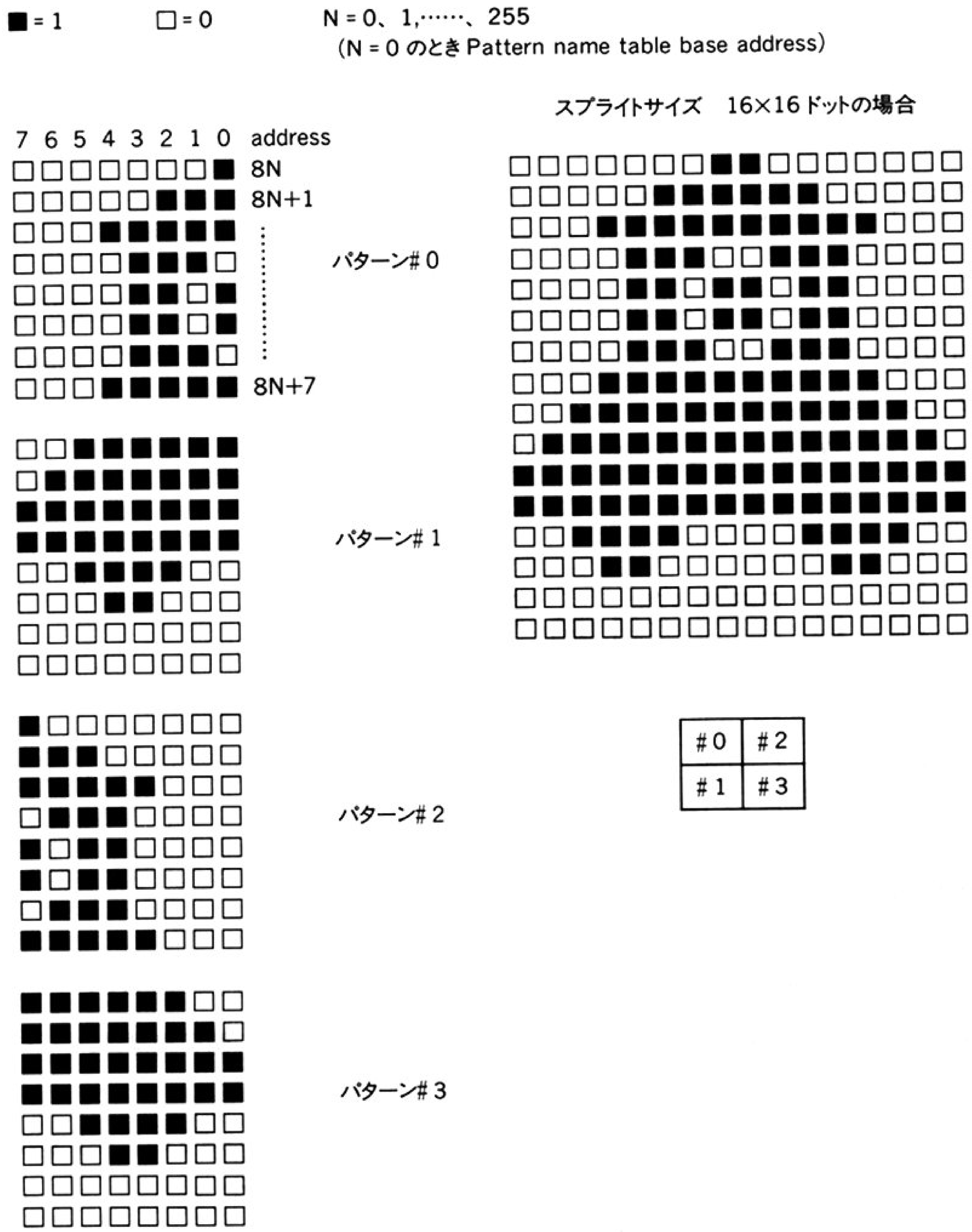


図 4.52　スプライトパターンジェネレータテーブルの例

スプライトのサイズを 16×16 とした場合、スプライトアトリビュートテーブルで指定するパターン番号は#0～#3 のどれでもかまいません。

6.2 スプライトモード2

スプライトモードはV9938で新たに追加された機能で、GRAPHIC 3〜7モード（SCREEN 4〜8）で使うことができます。

6.2.1 スプライトモード2の特徴

32個のスプライトには各々に#0〜#31の番号がつけられており、番号の若い方が優先度が高くなります。CRTの1水平線上には優先度の高い順に8個表示され、9番目以降のスプライトの重なった部分は表示されません。

スプライトモード2はTMS9918との互換性はありません。

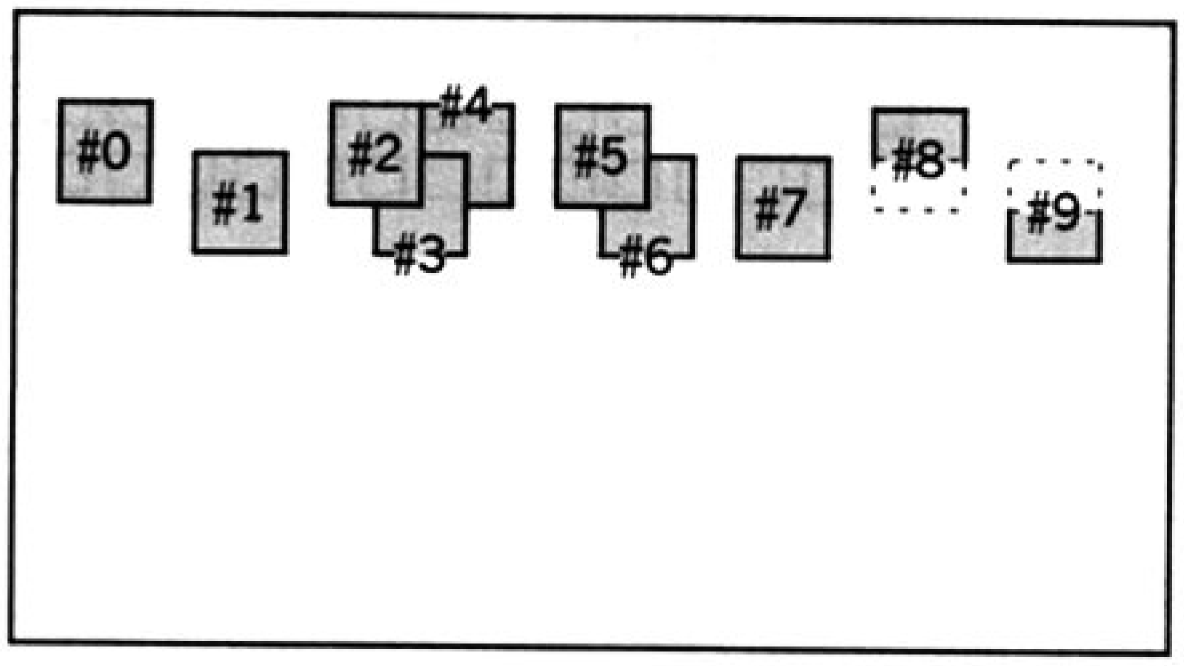


図4.53 スプライトモード2の最大表示数

2個のスプライトが衝突した（透明でない部分が重なった）ときは、ステータスレジスタS#0のビット5が1になるので、衝突の発生を知ることができます。このときの衝突座標は、ステータスレジスタ#3〜#5にセットされます。

1水平線上に9個以上のスプライトが並んだときは、ステータスレジスタS#0のビット6が1になり、S#0の下位5ビットに9番目のスプライトの番号がセットされます。

スプライトの色は1水平線ごとに指定できます。

スプライトカラーテーブルのCCビットを指定することによって、スプライトを組み合わせて使えます。これにより横方向の色をより多く指定することができます。

6.2.2 スプライトモード2の表示

スプライトを表示するには、次の操作を行います。

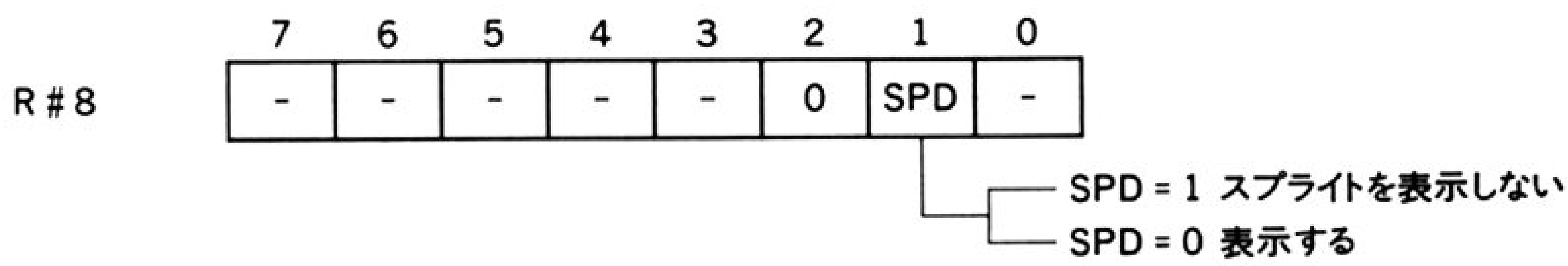
1) スプライトのサイズ

スプライトモード1と同じです。

2) スプライトの拡大

スプライトモード1と同じです。

3) スプライトの表示



4) スプライトパターンジェネレータテーブルのセット

VRAM上のスプライトパターンジェネレータテーブルに、スプライトのパターンをセットします(#0〜#255)。

5) スプライトカラーテーブルのセット

VRAM上のスプライトカラーテーブルに、スプライトの色、EC、CC、ICを1ラインずつセットします。

6) スプライトアトリビュートテーブルのセット

VRAM上のスプライトアトリビュートテーブルに、スプライトの座標、スプライトのパターン番号をセットします(#0〜#31)。

6.2.3 VRAM上の各テーブルの関係

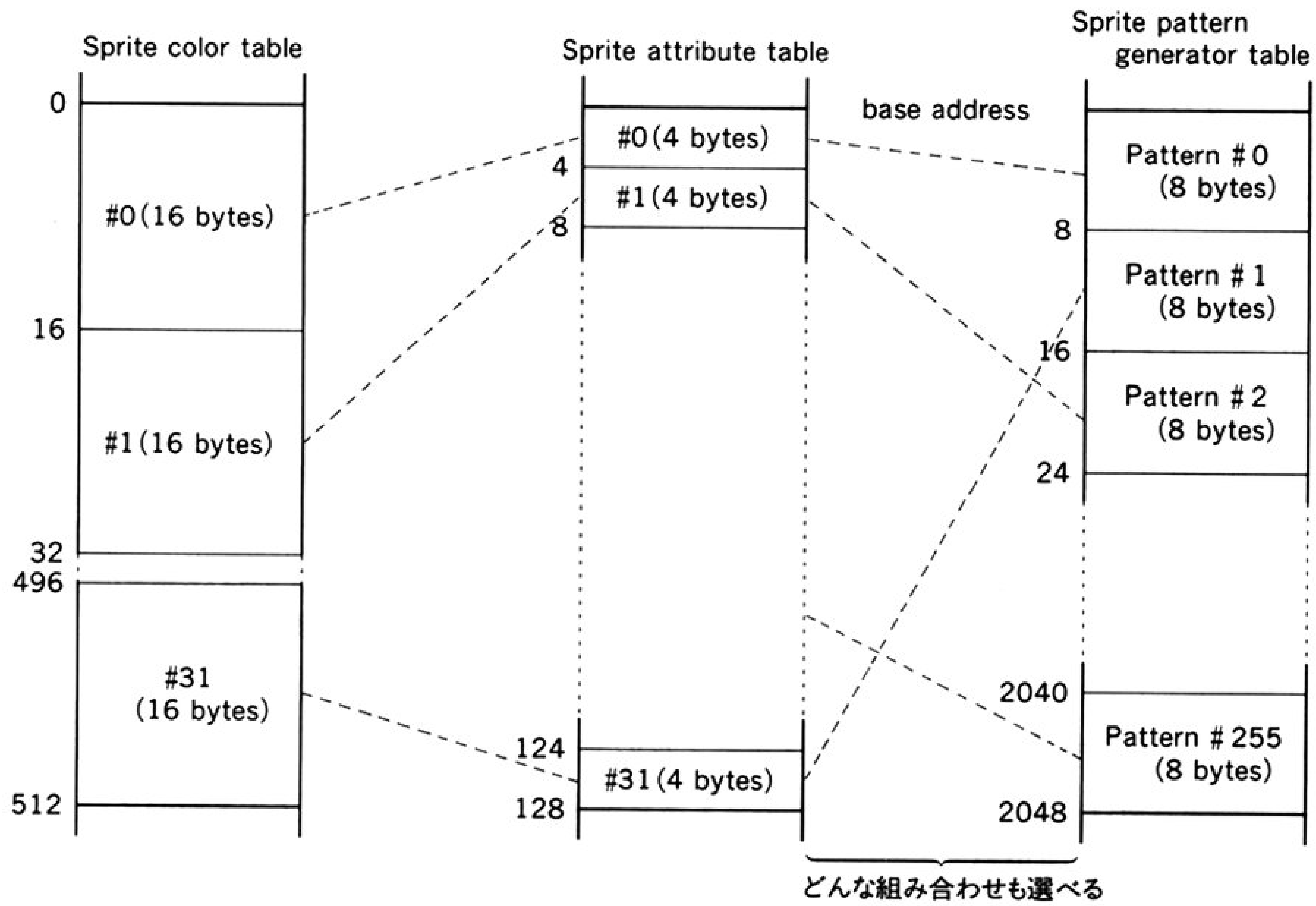


図4.54 VRAM上の各テーブルの関係

6.2.4 スプライトアトリビュートテーブル

32 個あるスプライトの各々の表示位置 (X、Y)、パターン番号を指定するための VRAM の領域です。スプライト 1 個につき 3 バイトのデータを書き込みます。



図 4.55 スプライトモード 2 のアトリビュートテーブル

1) Y 座標 (0〜255)

スプライトの Y 座標を指定します。

この Y 座標の値を 216 にすると、そのスプライトより優先度の低いスプライトは表示されません。たとえば# 10 のスプライトの Y 座標を 216 にすると、# 10〜# 31 のスプライトは表示されません。

2) X 座標 (0〜255)

スプライトの X 座標を指定します。

3) パターン番号 (0〜255)

スプライトパターンジェネレータテーブル上のパターン番号を指定します。スプライトのサイズが 16×16 のときは 4 個のパターン番号を 1 個のスプライトに使用しますが、このうちのどれを指定してもかまいません。

スプライトのサイズを 8×8 に指定した場合は 256 個のパターンを、16×16 に指定した場合は 64 個のパターンを指定することができます。

Sprite attribute table base address register

このビットは必ず1に設定します。

R#5	A14	A13	A12	A11	A10	A9	1	1
R#11	0	0	0	0	0	0	A16	A15

6.2.5 スプライトパターンジェネレータテーブル

スプライトモード1と同じです。

6.2.6 スプライトカラーテーブル

スプライトモード2では、32個のスプライトの各ラインごとにパターンの1の部分の表示色(0の部分は透明に固定)、優先順位の有無、衝突検出の有無、EC (Early Clock) の指定をすることができます。

スプライトカラーテーブルの先頭アドレスは、スプライトアトリビュートテーブルの先頭アドレスの値から512を引いた値に自動的にセットされます。

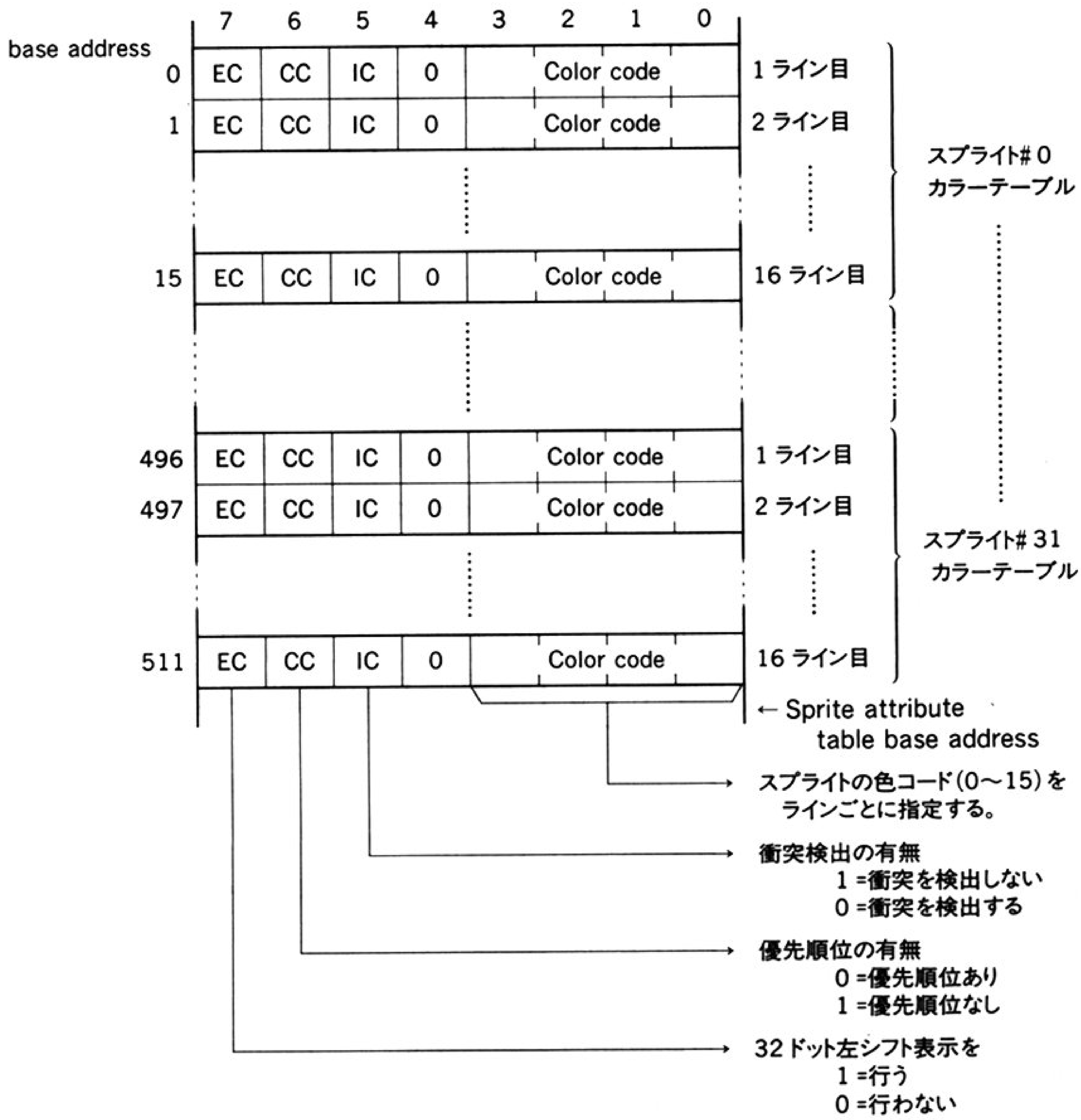


図4.56　スプライトモード2のカラーテーブル

6.2.7 スプライトの組み合わせ

スプライトモード2において、カラーテーブルのCCビットを「1」にすると、スプライトの優先順位をなくすことができます。

CCを「1」に指定した部分(1ラインごと)は、そのスプライトより若い番号で、かつCCが「0」のスプライトの存在する水平線上にのみ表示されます。この様子を図示したものが図4.57です。

このときも、スプライトが1水平線上に8個を超えると、9個目のスプライトは表示されません。

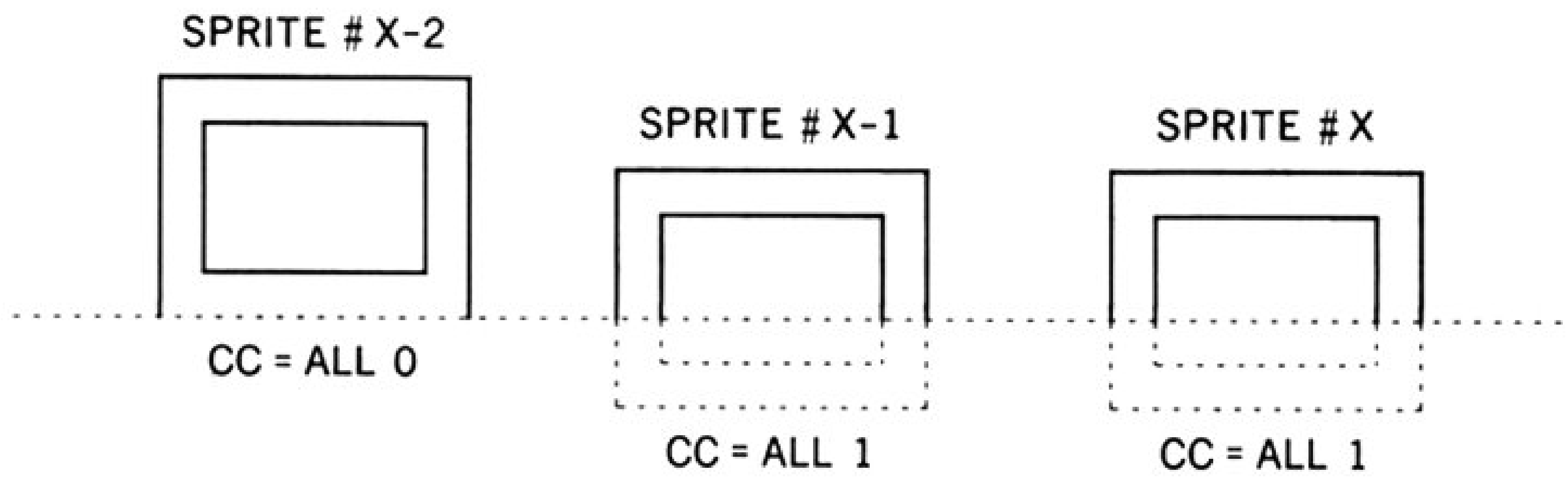


図 4.57　カラーテーブルの CC ビット

CC を「1」に指定した部分は、そのスプライトより若い番号で、かつ最もそのスプライトに番号が近いスプライトの CC = 0 の部分とは、たとえパターンが重なっても衝突の検出は行われません。また、重なった部分は各スプライトの色コードの OR をとって表示されます。

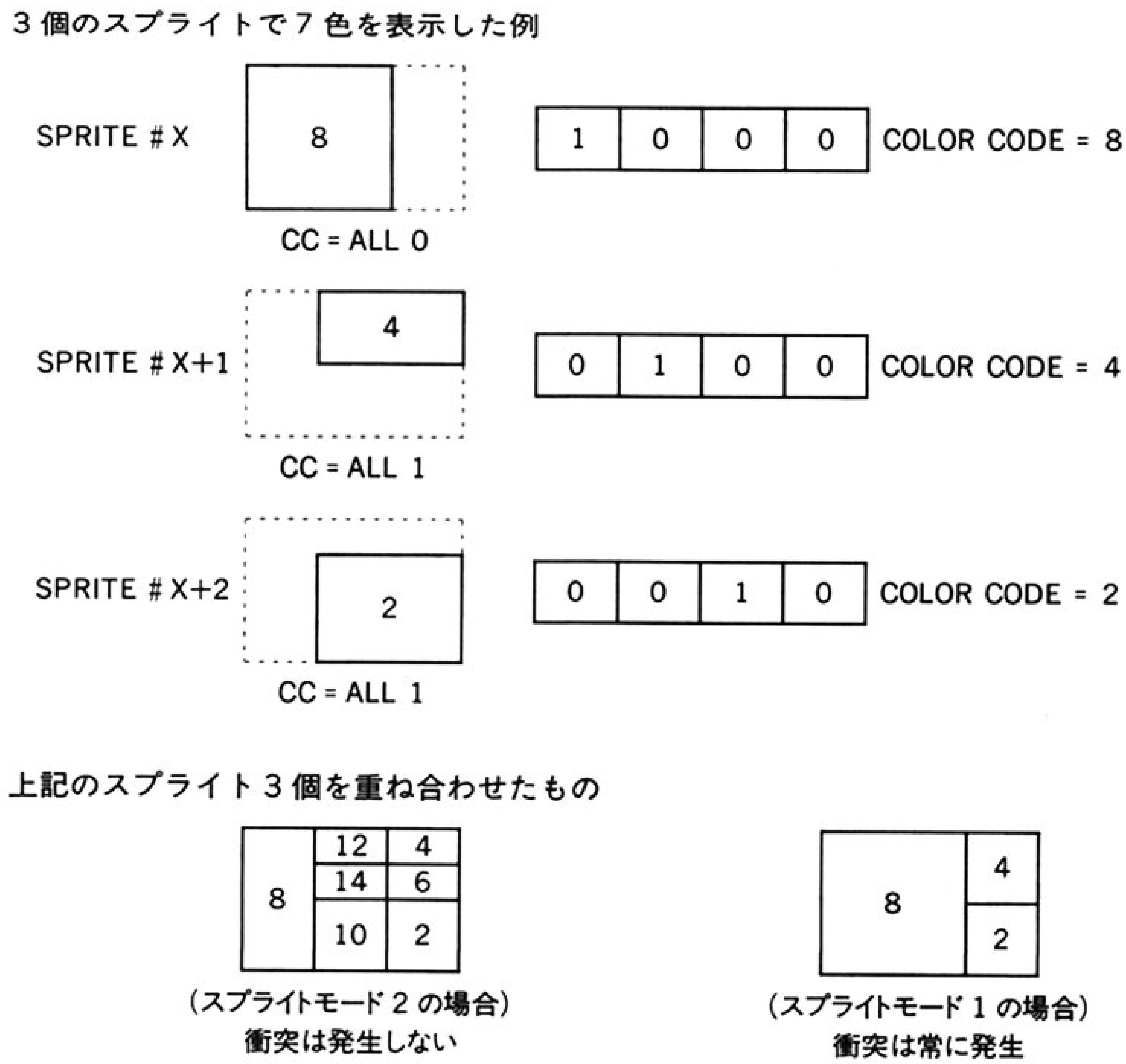


図 4.58　スプライトの重ね合わせ例

6.2.8 スプライトの衝突

スプライトの表示色が透明でなく、かつCCが「0」の部分が重なると衝突が発生します。衝突が発生するとステータスレジスタS#0のビット5が「1」になります。このビットはS#0を読み出すことによってリセットされます。

	7	6	5	4	3	2	1	0
S#0	-	-	C	-	-	-	-	-

C:
- 0 通常の状態
- 1 衝突が発生した

このとき、レジスタR#8のマウスフラグ(MS)またはライトペンフラグ(LP)がセットされていなければ(MSXではセットされることはない)、ステータスレジスタS#3〜S#6にスプライトの衝突座標がセットされます。

	7	6	5	4	3	2	1	0	
S#3	X7	X6	X5	X4	X3	X2	X1	X0	衝突X座標
S#4	1	1	1	1	1	1	1	X8	衝突X座標
S#5	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	衝突Y座標
S#6	1	1	1	1	1	1	Y9	Y8	衝突Y座標

ステータスレジスタS#3〜S#6は、S#5を読み出したときリセットされます。また、ステータスレジスタS#3〜S#6にセットされる値には、実際の衝突座標に対して次式のようなオフセットがつけられています。

実際の衝突座標(XC, YC)

X=XC+12　　Y=YC+8　　X=(S#4, S#3)

Y=(S#6, S#5)

6.3 スプライトの色設定

GRAPHIC 7 (SCREEN 8)以外の表示モードでは、スプライトの色コードは各モードの色コードと共通です。つまり、スプライトの画面上での表示色はパレットレジスタの値で決定されます。

GRAPHIC 7 (SCREEN 8) モードにおいてはスプライトの表示色は固定されており、パレットレジスタの影響を受けます。GRAPHIC 7 モードにおけるスプライトの色を表 4.7 に示します。

表 4.7 GRAPHIC 7 モード時のスプライトの色

COLOR CODE				GREEN			RED			BLUE		
C3	C2	C1	C0	G2	G1	G0	R2	R1	R0	B2	B1	B0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0	1	1	0	1	0
0	1	0	0	0	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0	0	1	0
0	1	1	0	0	1	1	0	1	1	0	0	0
0	1	1	1	0	1	1	0	1	1	0	1	0
1	0	0	0	1	0	0	1	1	1	0	1	0
1	0	0	1	0	0	0	0	0	0	1	1	1
1	0	1	0	0	0	0	1	1	1	0	0	0
1	0	1	1	0	0	0	1	1	1	1	1	1
1	1	0	0	1	1	1	0	0	0	0	0	0
1	1	0	1	1	1	1	0	0	0	1	1	1
1	1	1	0	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1

TP とスプライト

TP（レジスタ R#8のビット5）を操作することにより、カラーコード0は次のように定義され、スプライトもその影響を受けます。

TP＝0　カラーコード0は透明として扱われる
スプライトのカラーコード0で指定された部分は表示されず、他のスプライトと重なっても衝突は発生しません。

TP＝1　カラーコード0はパレットレジスタで指定された色になる（GRAPHIC7のときのみ、固定色R＝0、G＝0、B＝0にセットされる）
スプライトのカラーコード0で指定された部分は表示され、他のスプライトと重なると衝突を発生します。

7章 その他の機能

7.1 グラフィック画面 2 ページの交互表示

V9938 では、次の方法でグラフィック画面（GRAPHIC 4〜GRAPHIC 7）2 ページを交互に表示させることができます。交互に表示できるページの組み合わせは図 4.59 のとおりです。

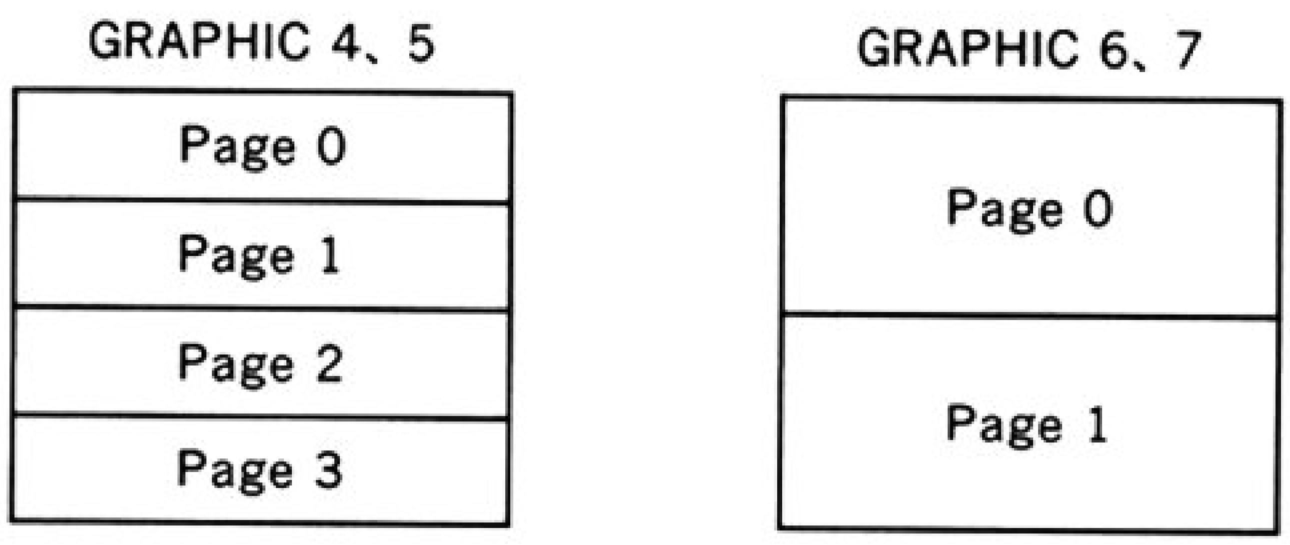


図 4.59　グラフィック画面の交互表示

7.1.1 レジスタ R#13 を使う方法

この方法によれば、各ページの表示時間をそれぞれ 166m 秒〜2053m 秒の間で指定できます。

1) パターンネームテーブルベースアドレス（レジスタ R#2）を奇数ページにセットする

2) レジスタ R#13 に ONTIME（偶数ページの表示時間）、OFFTIME（奇数ページの表示時間）をセットする（セットする値については、TEXT 2 の項を参照して下さい。）

Blinking period register

	7	6	5	4	3	2	1	0
R#13	ON3	ON2	ON1	ON0	OF3	OF2	OF1	OF0
	ON TIME				OFF TIME			

7.1.2 EOビットを使う方法

60Hz 周期で2ページを交互に表示させることができます。

1) パターンネームテーブルベースアドレス（レジスタ R#2）を、奇数ページにセットする

2) レジスタ R#9のビット2（EO）を「1」にセットする

Mode register 3

	7	6	5	4	3	2	1	0
R#9	-	-	-	-	-	EO	-	-

7.2 インタレース表示

V9938にはインタレース表示の機能があります。

7.2.1 第1フィールドと第2フィールドに同じページを表示させる場合

レジスタ R#9のILを「1」にセットする

7.2.2 第1フィールドに偶数ページ、第2フィールドに奇数ページを表示させる場合

1. レジスタ R#9のILを「1」に、R#9のEOを「1」にセットする
2. パターンネームテーブルベースアドレス（レジスタ R#2）を奇数ページにセットする

7.3 GRAPHIC5 モードでのスプライト

GRAPHIC 5 モードの静止画は水平方向の解像度が 512 であり、カラーコードは 2 ビット / ピクセルです。

スプライトは静止画の 1/2 の解像度で表示されます。すなわちスプライトパターンは水平方向 256 ピクセルの解像度で、座標系は X = 0〜255 となります。

図 4.60 のようなスプライトパターンは、図 4.61 のようにスプライトパターンジェネレータテーブルに書き込まれなければなりません。

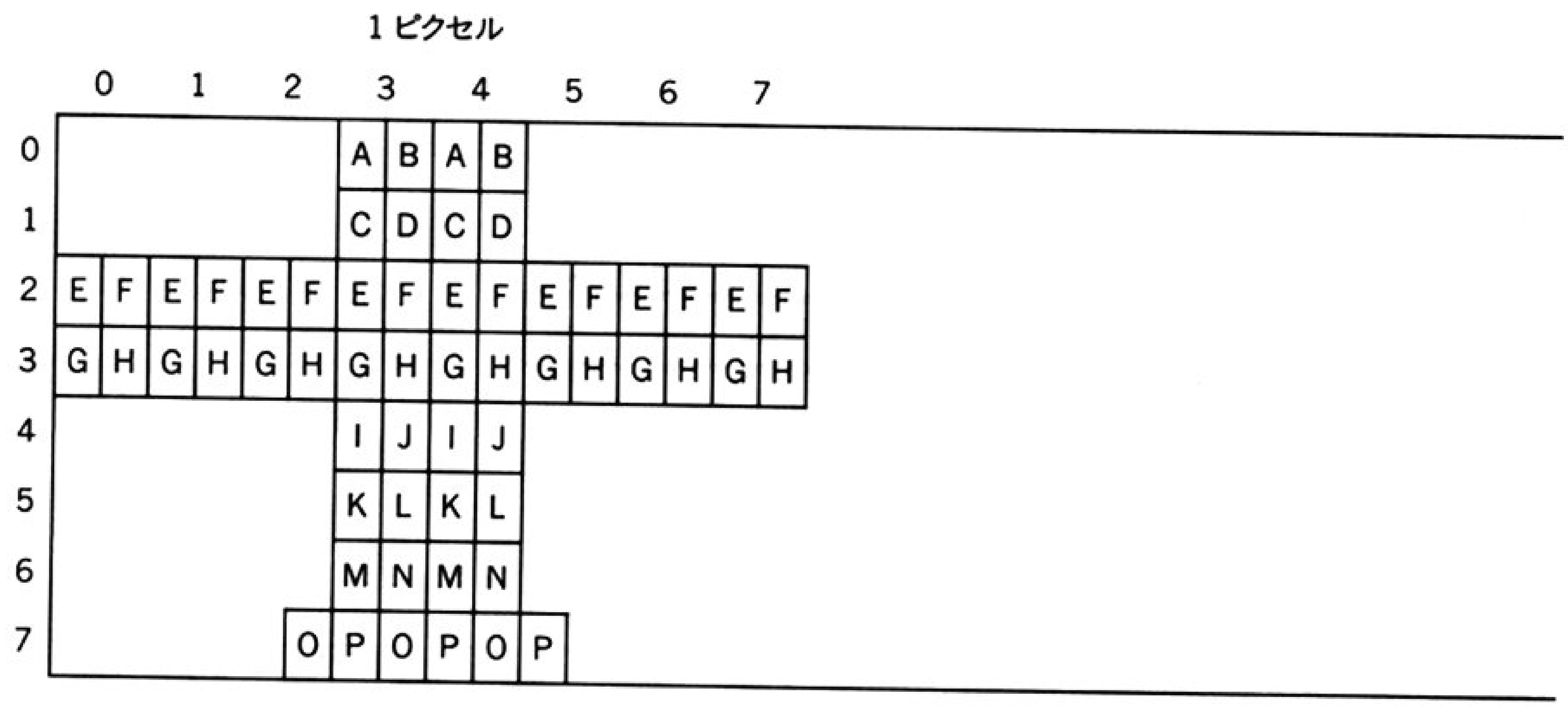


図 4.60　GRAPHIC 5 モードでのスプライトパターン例

0	0	0	0	1	1	0	0	0
1	0	0	0	1	1	0	0	0
2	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1
4	0	0	0	1	1	0	0	0
5	0	0	0	1	1	0	0	0
6	0	0	0	1	1	0	0	0
7	0	0	1	1	1	1	0	0

図 4.61　GRAPHIC 5 モードでのパターンジェネレータテーブル例

スプライトの色はスプライトカラーテーブルの内容によって決まります。カラーテーブルの上位4ビットはEarly Clockなどのコントロールとして使われます。下位4ビットは2ビットずつに分けられ、上位側の2ビットでスプライトの偶数ピクセルの色、下位側の2ビットで奇数ピクセルの色を表わします。このとき、スプライト色づけの解像度は512です。

図4.62のようにカラーテーブルに書き込まれている場合に、図4.60のように表示されます。

0		A	B
1		C	D
2		E	F
3		G	H
4		I	J
5		K	L
6		M	N
7		O	P

図4.62　GRAPIHC 5モードでのカラーテーブル例

7.4 同期モードの選択

V9938では、同期モードはレジスタR#9のS1とS0の2ビットで指定します。

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	E0	$\overline{NT}$	DC

表4.8　同期モード

S1	S0	同期モード	$\overline{Ys}$	用　途
0	0	パソコン同期	常にV9938を選択（0）	V9938の画面を表示
0	1	標準同期	表示画面の透明部分で発生	スーパーインポーズ、デジタイズなど
1	0	標準同期	常に外部信号を選択	外部信号の表示
1	1	予約	予約	予約

8章 V9958の構成

「V9958」は、MSX2＋用のVDP（Video Display Processor）として開発されたVLSIで、TMS9918A（MSXのVDP）およびV9938（MSX2のVDP）とソフトウェア上位互換性があります。V9958は株式会社アスキーとヤマハ株式会社によって共同開発され、V9938に加えて以下の特徴があります。

- YJK方式により最大19,268色を同時に表示可能
- ハードウェアによる水平スクロール
- 全表示モードでコマンド機能を使用可能

この章ではV9938に対して変更・追加された機能を解説します。この章で説明のない部分についてはV9938と同じ仕様です。

8.1 レジスタ

8.1.1 追加されたレジスタ

	7	6	5	4	3	2	1	0
R#25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2
R#26	0	0	HO8	HO7	HO6	HO5	HO4	HO3
R#27	0	0	0	0	0	HO2	HO1	HO0

注意　空ビットには、必ず「0」を設定してください。
リセット時には、すべてのビットは「0」に設定され、機能的にはV9938と同等になります。

8.1.2 変更されたレジスタ

Status register 1のID番号を「2」とし、V9958であることを表します。V9938では「0」が返ってきます。

	7	6	5	4	3	2	1	0
S#1	FL	LPS	0	0	0	1	0	FH

ID#

8.2 V9958の機能

V9958ではV9938を拡張して、新たな機能を加えています。追加された機能には以下のようなものがあります。

8.2.1 水平スクロール

	7	6	5	4	3	2	1	0
R#25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2
R#26	0	0	HO8	HO7	HO6	HO5	HO4	HO3
R#27	0	0	0	0	0	HO2	HO1	HO0

R#25 MSK　0=左側8ドットはマスクされない(初期値)

1=左側8ドットをマスクし、ボーダーカラーを表示する

R#27の値が「0」の場合は、マスクする必要はありません(GRAPHIC 5、6では16ドットのマスクになる)。

SP2　0=水平方向画面サイズを1ページとする(初期値)

スクロールは1ページ内で行われ、画面上には左側の隠れた部分が右側に表れます。

1=水平方向画面サイズを2ページとする

スクロールは2ページで行われ、スクロールによって裏のページが表れます。

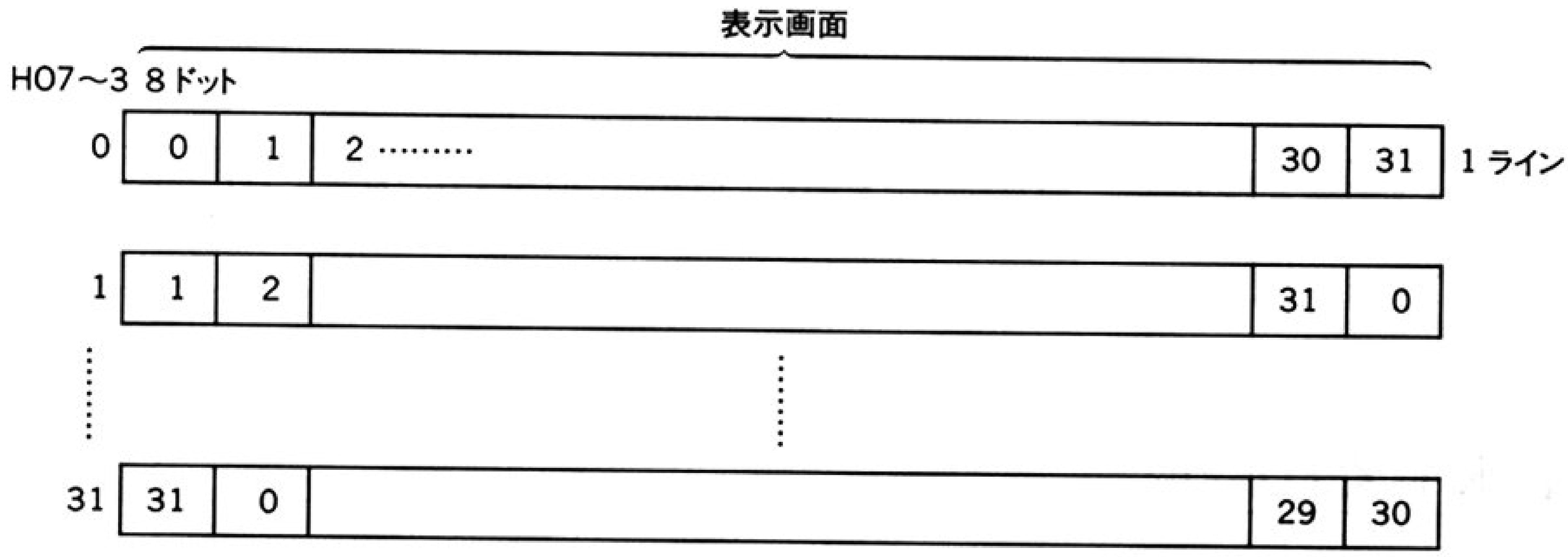
HO8～HO0 静止画の水平方向スクロール量をドット単位で設定する（GRAPHIC 5、6 は 2 ドット単位のスクロールになる）

水平スクロール時の VRAM アクセスは、8 ドット単位で行われ、R # 27 の値が「0」以外のときは、表示用のデータが不定になります。この不定のデータを表示させないためにマスクが必要になります。

HO8～HO3 に対する表示画面

左方向へ、8 ドット単位（GRAPHIC 5、6 は 16 ドット単位）で設定値だけシフトします。

SP2 = 0 のとき



注　意　HO8 は無視されます。

SP2 = 1 のとき

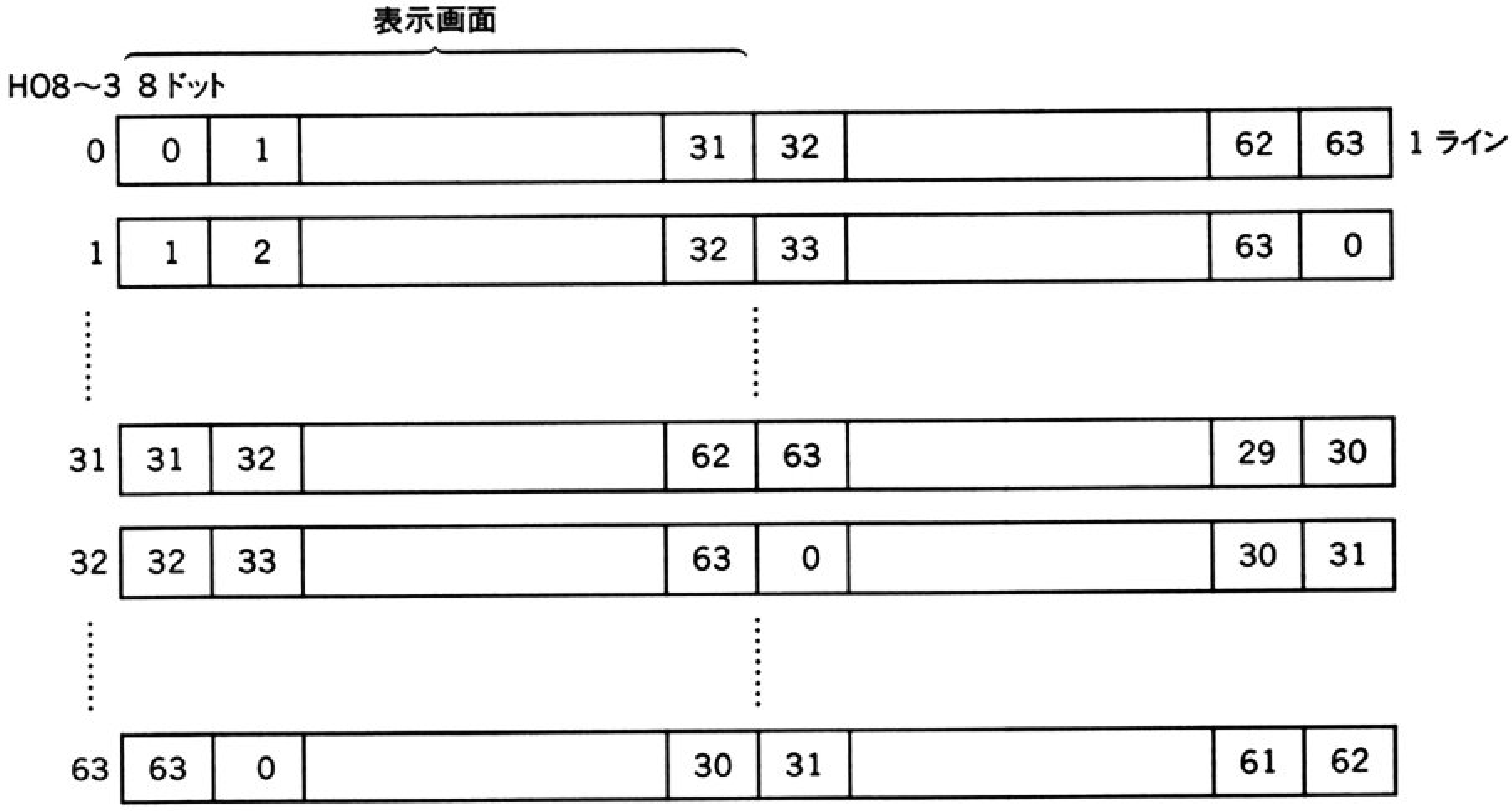


図 4.63 HO8～HO3 に対する表示画面

注 意	SP2 = 1 のときは、パターンネームテーブルの先頭アドレスは、奇数ページが選択されるように設定します。 各テーブルのベースアドレスは次のようになります。

パターンネームテーブル(PNT)	0～31 偶数ページ 32～63 奇数ページ
パターンジェネレータテーブル(PGT)	ベースアドレス設定値 スクロールによって変化しない
カラーテーブル(CT)	ベースアドレス設定値 スクロールによって変化しない

HO2～HO0 に対する表示画面

右方向へ、1ドット単位（GRAPHIC 5、6 は、2ドット単位）で設定値だけシフトします。

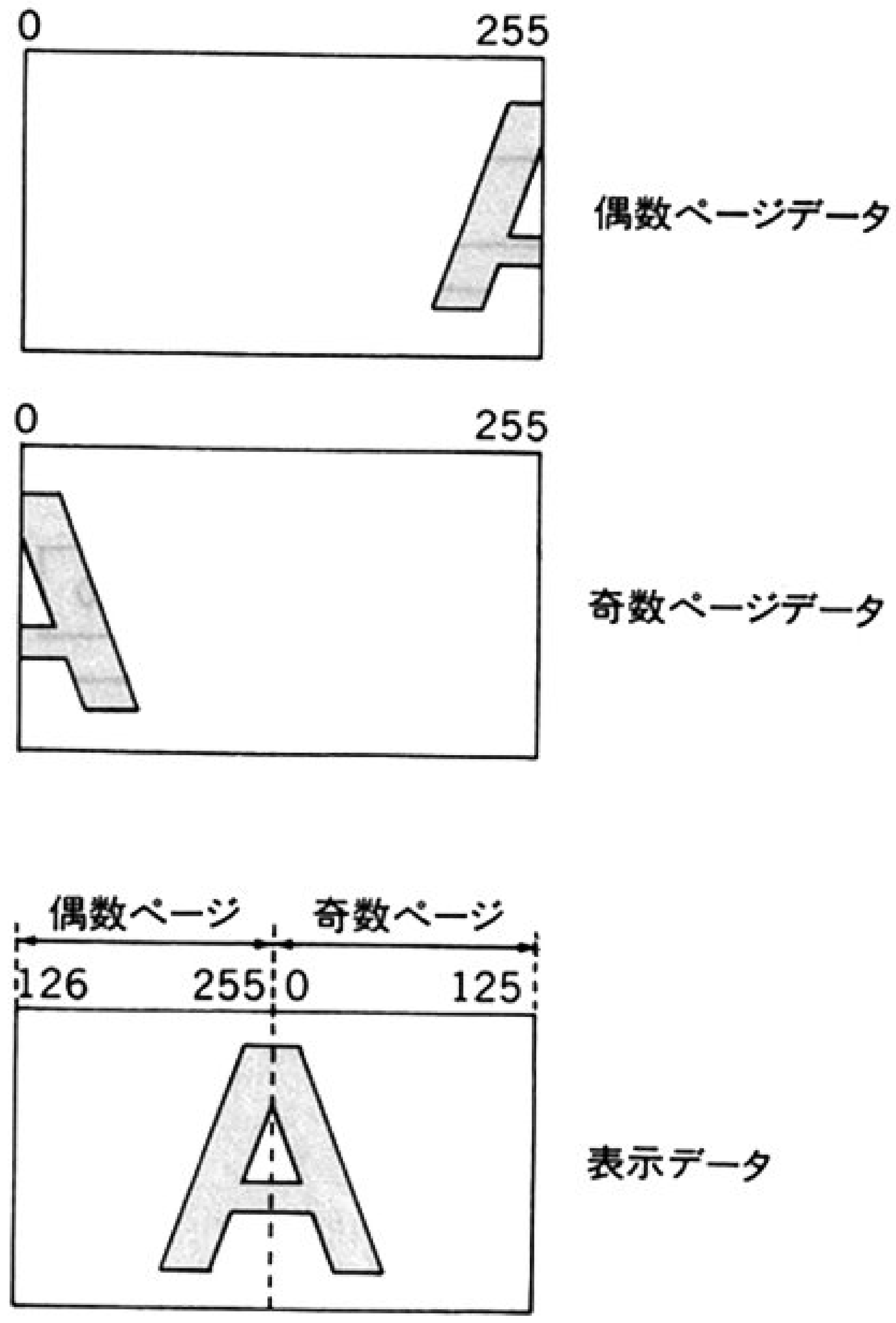


図 4.64　2 画面水平スクロールの例

図 4.64 のように、左へ 126 ドットスクロールするときは、以下のようになります。

R＃26＝128＝8×16　(左へ 128 ドット)
R＃27＝2　(右へ 2 ドット)

8.2.2 ウエイト機能

	7	6	5	4	3	2	1	0
R＃25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2

WTE　0＝ウエイト機能を無効にする (初期値)
V9938 と同様に機能します。
1＝ウエイト機能を有効にする
CPU が VRAM をアクセスした際に、V9958 の VRAM アクセスが完了するまで、全ての V9958 ポートへのアクセスに対してウエイト信号を発生します。

レジスタとカラーパレットへのアクセス未完了およびコマンドのデータレディによるウエイト機能はありません。

注　意　ウエイト機能はMSXでは使っていません。
このビットを「1」にするときは、その前にダミーのVRAMアクセスを行って下さい。

8.2.3 コマンド機能

	7	6	5	4	3	2	1	0
R#25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2

CMD　0＝GRAPHIC 4～7の表示モードでのみコマンド機能を有効にする（初期値）
1＝全表示モードにおいてコマンド機能を有効にする
GRAPHIC 4～7モードでは、V9938と同様に機能し、それ以外の表示モードでは、GRAPHIC 7モードとして動作します。したがって、GRAPHIC 4～7モード以外でコマンドを使用するときには、パラメータはGRAPHIC　7モードのX、Y座標系で設定します。

8.2.4 YJK方式データの表示機能

	7	6	5	4	3	2	1	0
R#25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2

YJK　0＝VRAM上のデータをRGB方式。（例えば、GRAPHIC 7のときは各3、3、2ビット）として扱う（初期値）。
スプライトの表示色は従来どおりです。
1＝VRAM上のデータをYJK方式とみなし、これをRGB信号（各5ビット）に変換し、アナログ出力する。
スプライトの表示色にはパレットが有効になります。

YAE　YJK方式のデータフォーマットの選択

以下は、YJK方式のデータフォーマットです。連続した4ドットをグルーピングして表わします。

1）アトリビュートがない場合（YAE＝0）（初期値）

	C7	C6	C5	C4	C3	C2	C1	C0
1ドット	Y_1					KL		
1ドット	Y_2					KH		
1ドット	Y_3					JL		
1ドット	Y_4					JH		

□は1ドット分のカラー情報です。

図4.65　アトリビュートなしのYJKデータフォーマット

YJK方式のデータは連続した4ドット分の情報でグループ分けされており、

Y_1、KL、KH、JL、JHで先頭1ドットのカラー情報
Y_2、KL、KH、JL、JHで2ドット目のカラー情報
Y_3、KL、KH、JL、JHで3ドット目のカラー情報
Y_4、KL、KH、JL、JHで4ドット目のカラー情報

となります。

KとJの値は4ドットに共通で使われます。Yの値は1ドットごとに指定します。

2）アトリビュートがある場合（YAE＝1）

	C7	C6	C5	C4	C3	C2	C1	C0
1ドット	Y_1				A	KL		
1ドット	Y_2				A	KH		
1ドット	Y_3				A	JL		
1ドット	Y_4				A	JH		

Aはアトリビュート

図4.66　アトリビュートありのYJKデータフォーマット

A＝0　　YAE＝0の場合と同じ（初期値）
Y、J、Kは全てYJK方式のデータとなります。□は1ドット分のカラー情報です。Aのビットは無視されます。

A＝1　　Y1〜Y4はそれぞれカラーコードとなり、カラーパレットを通してRGB出力されます。JとKはYJK方式のデータとなります。

3) YJK と JAE の組み合わせ

YJK	YAE	VRAM データ
0	×	従来のカラーパレット経由
1	0	YJK → RGB 変換テーブル経由
	1	A = 0 ……YJK → RGB 変換テーブル経由 A = 1 ……カラーパレット経由

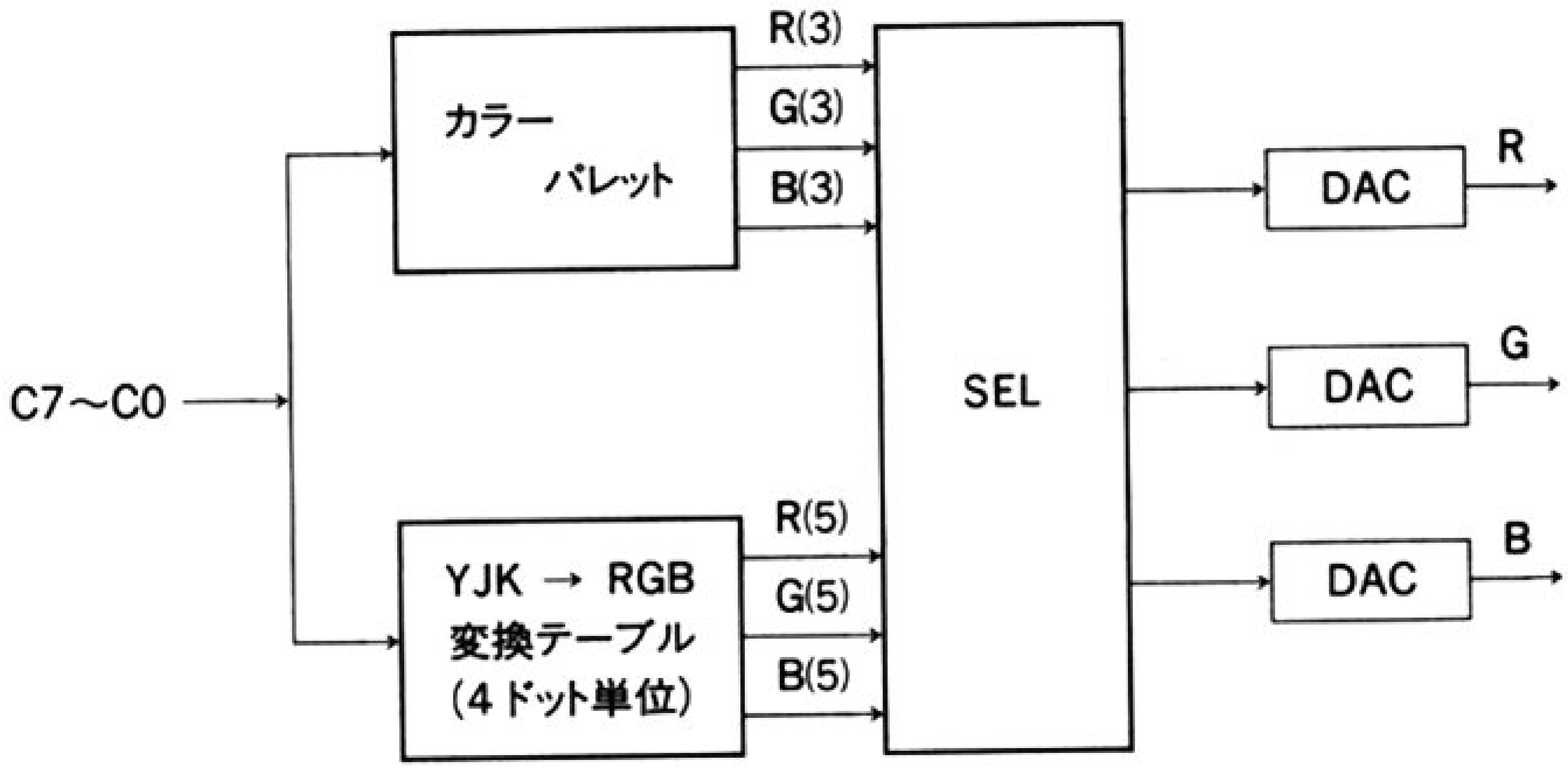


図 4.67 YJK と JAE の組み合わせ

8.2.5 V9958 では削除された機能

1. コンポジットビデオ出力
2. マウス・ライトペンインターフェイス

この変更に伴い、次の内部レジスタのビット（■の部分）が意味を持たなくなります。これらのレジスタに値を書き込む場合には、これらのビットに必ず「0」を書き込んで下さい。

Status Register 1

	7	6	5	4	3	2	1	0
S # 1	FL	LPS	0	0	0	1	0	FH

FL　ライトペン光検出フラグ

LPS　ライトペン・マウススイッチ

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	M5	M4	M3	0

IE2　ライトペンによる割込み
(MSX2 は V9938 のライトペン機能は使用していない)

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

MS　マウスで使用

LP　ライトペンで使用
(MSX2 は V9938 のマウス・ライトペン機能は使用していない)

9章 YJK方式

9.1 YJK方式とは

MSX2までは、色の情報を扱うために、色を赤、緑、青の信号に分解してその割合を数値で表現する方式(RGB方式)が採用されていました。これに対して、YJK方式とは色情報を輝度成分(Y)と色相成分(JとK)に分解してVRAM上に置き、表示するときにRGB方式に変換するものです。

V9958ではVRAMの容量をV9938と同じにするために、輝度情報については、画面1ドットに対して5ビット(0〜31)設定できるようになっていますが、色相情報については、画面4ドットに対して6ビット(−32〜+31)しか設定できません(SCREEN 12の場合)。

YJK方式は、デジタイズ処理の際も4ドットごとのJ、Kの値をどうするかなどの点で難しい面がありますが、最大の特徴はVRAM容量を変えずに色数が増えたことです。YJK方式は、ほとんど共通のRとGをJKとし、これにBおよび明るさのグラデーションを付けるYを加えて表示する方式なので、同じ色調のグラデーション(自然界に多い)や、白や灰色などの無彩色の背景上にある物体などを表現する能力が高く、真っ青な背景に斜めの明るい黄色や赤の細い線などの人工的な物体の表現能力はあまり高くないという性質があります。つまり、人工的なコンピュータグラフィックスなどには向かない場合もありますが、自然物の表示には相当の効果があります。

19,268色という数字は、このYJKの取り得る全ての値で導かれるRGBの組み合せを計算して、重複しているものを除いたものです。

9.2 YJK と RGB の変換

9.2.1 YJK から RGB への変換式

$$R = Y + J$$

$$G = Y + K$$

$$B = \frac{5}{4}Y - \frac{1}{2}J - \frac{1}{4}K$$

9.2.2 RGB から YJK への変換式

$$Y = \frac{1}{2}B + \frac{1}{4}R + \frac{1}{8}G$$

$$J = R - Y$$

$$K = G - Y$$

10章
V9958の画面モード

MSX BASIC 3.0では、YJKモードおよびYJK/RGB混在モード(以下、YJKモードとYJK/RGB混在モードを総称して、自然画モードと呼ぶ)として、SCREEN 10～12を設けていますが、V9958ではそれに相当するGRAPHICモードはありません。

V9958を自然画モードにするためには、表示モードをGRAPICH 7にした上で、R#25のbit 3、4を操作します。詳しくは、10.4のサンプルプログラムをご参照下さい。

10.1 SCREEN 9

SCREEN 9は韓国バージョンのMSXでハングル文字を表示するために使われています。したがって、日本製のMSXでは使用しません。

10.2 SCREEN 10、11

10.2.1 特徴

■解像度	横256×縦212ドット（または縦192ドット）の YJK・RGB混在モード
■表示色	12,499色同時（全画面）
■スプライト	モード2（パレットが使用可能）
■1画面に必要なVRAM容量	128Kバイト（2画面）

10.2.2 関係するレジスタとVRAMの領域

GRAPHIC 7（SCREEN 8）と同じです。BASICではSCREEN 10またはSCREEN 11として扱います。

10.2.3 初期設定

R#25以外は、GRAPHIC 7（SCREEN 8）と同じです。

	7	6	5	4	3	2	1	0
R#25	0	CMD	VDS	1	1	WTE	MSK	SP2

□は表示モード設定用のビットをYJK・RGB混在モードにセットした例です。この表示モードでは、LN=1のとき縦212ドット、LN=0のとき縦192ドット（BASICではサポートしない）に設定されます。その他のビットは任意に設定します。

1) パターンネームテーブルの設定

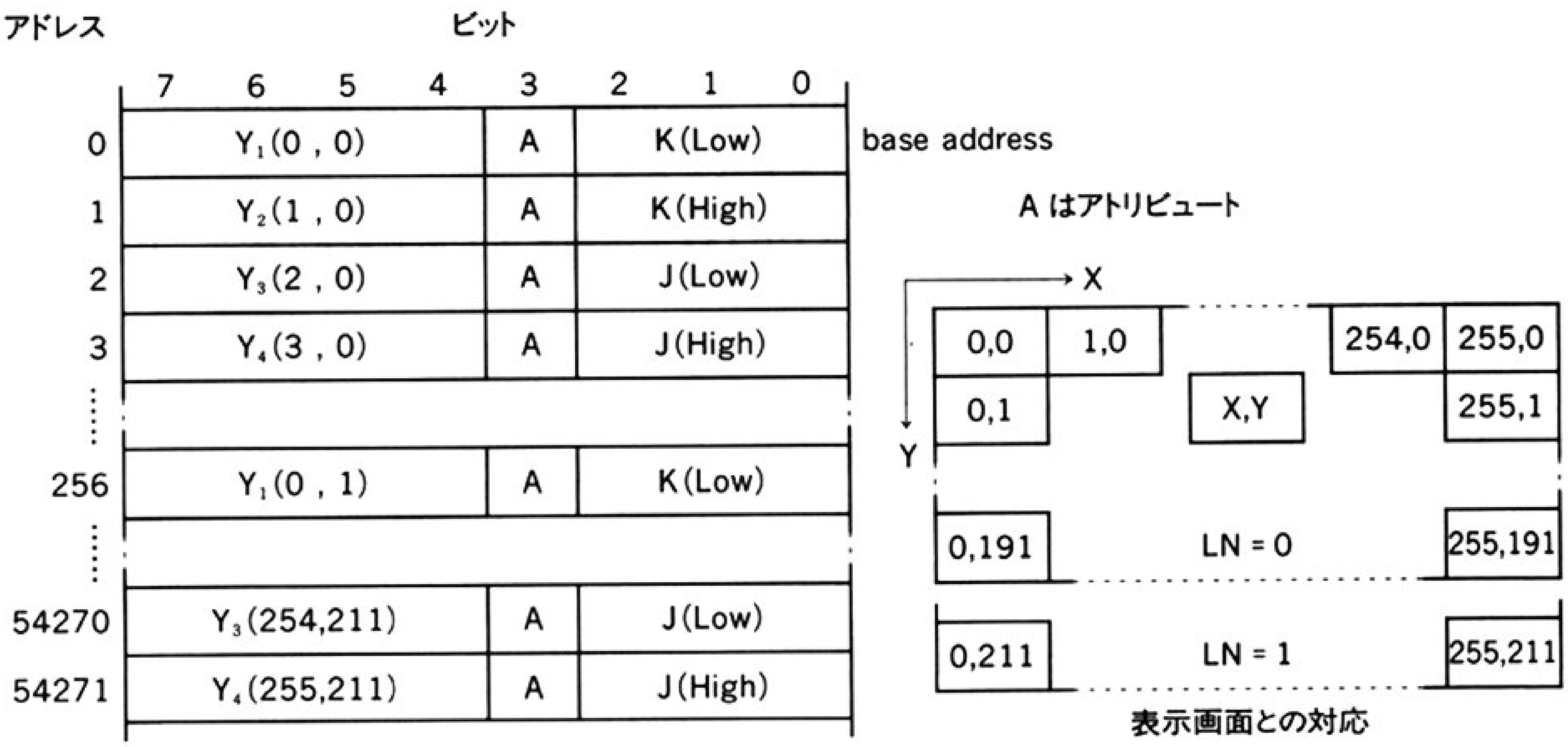


図 4.68　SCREEN 10、11 モードのパターンネームテーブル

10.2.4 SCREEN 10、11 モードの VRAM マップ

GRAPHIC 7（SCREEN 8）と同じです。

10.2.5 SCREEN 10とSCREEN 11の違い

VDP 上では SCREEN 10 と 11 は何の違いもありません。アトリビュートが「0」の画素は YJK 方式で表示され、アトリビュートが「1」の画素は Y の 4 ビットが色コードとして扱われます。MSX-BASIC 3.0ではこのYJK / RGB混在モードを更に2つのスクリーンモードとして扱います。

SCREEN 10 では、

- LINE、PSET などの描画コマンド
- PRINT #文による文字表示
- 漢字モードにおけるグラフィック画面への PRINT 文

などで色指定を行うときは、SCREEN 0 から 7 までで使用する通常の色コードである 0～15 だけが使用できます。これ以外の値は Illegal function call となります。この色指定で RGB 以外の部分、すなわち JK の成分（下位 3 ビット）はそのままにして、描画されない画素の表示が影響され

ることを防ぎます。これによって、背景に自然画をロードしておいてその上に線を引くようなことが可能となります。

例えば、色コード15のドットはこの画面ではY成分の部分（上位4ビット）およびアトリビュートビットを1にすることになり、VRAM上では&HF8となりますが、SCREEN 10のときは描画に際して、このような変換を意識する必要はありません。

一方、SCREEN 11では、色コードの扱いはVRAM上の値そのものとなり、色コードは0〜255を指定できます。

10.3 SCREEN 12

10.3.1 特徴

■解像度	横 256×縦 212 ドット（または縦 192 ドット）の YJK モード
■表示色	19,268 色同時（全画面）
■スプライト	モード 2（パレットが使用可能）
■ 1 画面に必要な VRAM 容量	128K バイト（2 画面）

10.3.2 関係するレジスタと VRAM の領域

GRAPHIC 7（SCREEN 8）と同じです。BASIC では SCREEN 12 として扱います。

10.3.3 初期設定

R # 25 以外は、GRAPHIC 7（SCREEN 8）と同じです。

	7	6	5	4	3	2	1	0
R # 25	0	CMD	VDS	0	1	WTE	MSK	SP2

▨は表示モード設定用のビットを YJK モードにセットした例です。この表示モードでは、LN = 1 のとき縦 212 ドット、LN = 0 のとき縦 192 ドット（BASIC ではサポートしない）に設定されます。その他のビットは任意に設定します。

1) パターンネームテーブルの設定

アドレス	ビット 7 6 5 4 3	2 1 0	
0	Y_1(0 , 0)	K(Low)	base address
1	Y_2(1 , 0)	K(High)	
2	Y_3(2 , 0)	J(Low)	
2	Y_4(3 , 0)	J(High)	
⋮			
256	Y_1(0 , 1)	K(Low)	
⋮			
54270	Y_3(254,211)	J(Low)	
54271	Y_4(255,211)	J(High)	

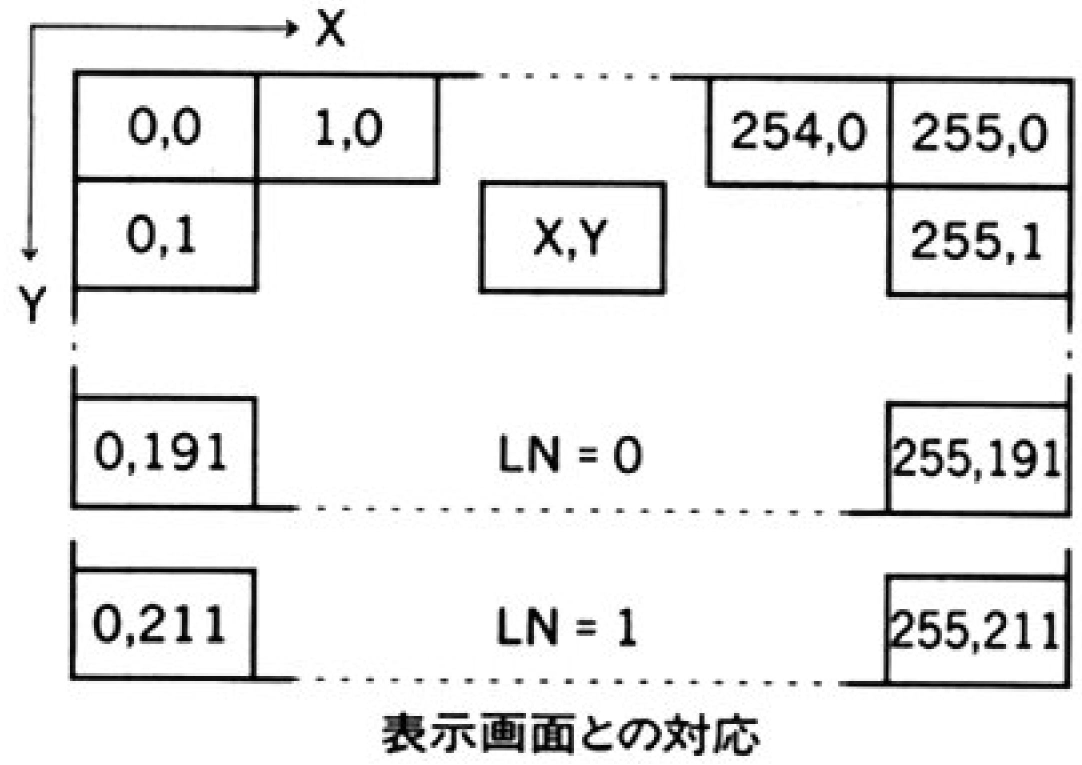


図 4.69　SCREEN 12 モードのパターンネームテーブル

10.3.4 SCREEN 12 モード VRAM マップ

GRAPHIC 7（SCREEN 8）と同じです。

10.4 SCREEN 10、11、12 の設定方法

SCREEN 0～8 の切り換えは、モードレジスタに値を設定することによって行いますが、自然画モード (SCREEN 10～12) については、別の方法を使います。なぜならば、自然画モードとはV9958 にとって、GRAPHIC 7 モードで R#25 の YEA ビットと YJK ビットを設定した状態だからです。

したがって、GRAPHIC モードが 7 の状態で以下のプログラムを実行すれば、SCREEN モードを 10、11、12 に設定することができます。

MODE 変数の bit 5 の SET、RESET は ROM 内のルーチン (GRPPRT、NVBXLN など) を使用する可能性がある場合にのみ必要で、アプリケーションが VDP を直接操作する場合には必要ありません。もちろん、そのような場合には、アトリビュートの操作はアプリケーションプログラムから行わなければなりません。

また、BASIC で行っているようなサービス (SCREEN12 から SCREEN10 に切り換えた時アトリビュートを落としている) は、以下の中には含まれていません。

リスト 4.1　SCREEN 10、11、12 の設定方法

```
mode     equ     0fafch
rg25sa   equ     0fffah          ; register #25 save area
wrtvdp   equ     47h

screen10:
         ld      hl,mode
         res     5, (hl)
         ld      a, (rg25sa)
         or      00011000b
         ld      b,a
         ld      c,25
         jp      wrtvdp

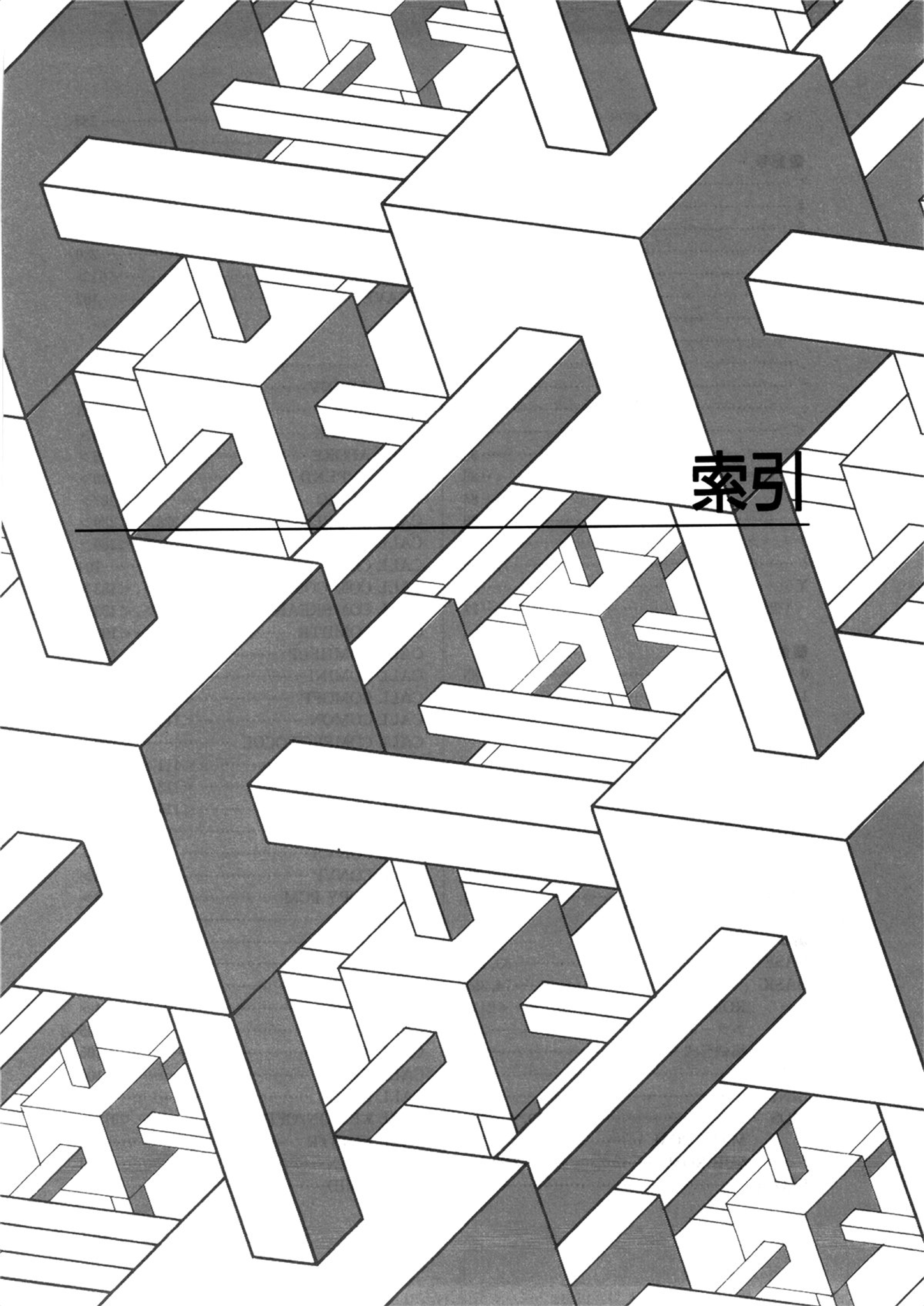
screen11:
         ld      hl,mode
         set     5, (hl)
         ld      a, (rg25sa)
         or      00011000b
         ld      b,a
         ld      c,25
         jp      wrtvdp

screen12:
         ld      a, (rg25sa)
         and     11100111b
         or      00001000b
         ld      b,a
```

```
        ld      c,25
        jp      wrtvdp

        end
```

索引



(<　>内は Volume 2 のページ数を示します)

■記号

■数字

■A

■B

■C

■ D

参考文献

「MSX テクニカルデータブック 1　増補改訂版」　アスキー

「MSX2 テクニカルハンドブック」　アスキー

「V9938 MSX-VIDEO テクニカルデータブック」　アスキー

「YM2413 アプリケーションマニュアル」　ヤマハ

「Y8950 アプリケーションマニュアル」　ヤマハ

お問い合わせについて

弊社では厳重に梱包した上、細心の注意を払って製品を発送しております。万一、輸送上のトラブルが起こった場合には、ご一報いただければ新しいものと交換いたします。

マニュアルの作成については、なるべく詳細な説明をするよう心がけたつもりですが、理解できないところは、実際にコンピュータと向き合って納得のいくまで確かめて下さい。また、他のページを参照するのもひとつの方法です。それでも疑問点が解決できないときは、封書にて、下記の要領でお問い合わせ下さい。恐れ入りますが、このパッケージの性格上、お電話でのお答えは不可能と存じますので、ご了承下さいますようお願い申し上げます。

記

1. 送付先　〒107-24 東京都港区南青山6-11-1　スリーエフ南青山ビル
   株式会社アスキー　ユーザーサポート係
2. 必要項目
   (1) お客様の氏名、郵便番号、住所、電話番号（市外局番も含む）
   (2) 製品名、ユーザーID番号、製品シリアル番号
   (3) 機器構成
   本体のメーカー名、機種名
   接続している周辺機器のメーカー名、機種名（型番）
   使用しているソフトウェアのメーカー名とその名前
3. お問い合わせ内容

お問い合わせの内容は、製品のマニュアルに記述されている用語を用いて、具体的かつ明確に記述して下さい。なお、障害と思われる現象については、その現象を再現可能な情報が必要です。当社で再現できないものは調査できません。その現象が発生するまでの操作手順、データを必ず添付して下さい。データディスクがある場合は、そのコピーも同封して下さい。

なお、お客様がプログラミングされたアプリケーションソフトウェアの設計、作成、運用、保守などについては、当社のサポート範囲外ですので、お問い合わせいただいても回答できません。例えば、「このプログラムリストはなぜ動かないのか」といった質問にはお答えできません。

また、MSX-MUSICとMSX-AUDIO用のFM音源LSIの技術資料は、ヤマハ株式会社のご好意により掲載させていただきましたが、この内容についての問い合わせも弊社で承ります。ただし、具体的な音作りなどに関するご質問にはお答えできませんので、ご了承下さいますようお願い申し上げます。

MSX-Datapack Volume 1

1991年2月1日　第1版第1刷

編　　集　株式会社アスキー　システム事業部

発 行 所　株式会社アスキー
担当　遠藤　祥、北浦　訓行
〒107-24 東京都港区南青山6-11-1 スリーエフ南青山ビル
TEL (03)3486-7111(大代表)

制　　作　株式会社ジャパックスインターナショナル

ASCII